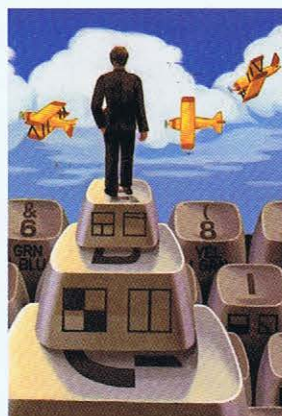
INTRODUZIONE AL BASIC

Parte 1

Andrew Colin

Corso di auto-apprendimento al linguaggio Basic per Commodore 16 e Plus/4

C16 plus/4 INTRODUZIONE AL BASIC Parte 1

T 1624

Questo corso è la Parte 1 di una serie intesa ad aiutare ad imparare qualsiasi aspetto della programmazione del Computer **16** e **PLUS/4**. Il presente corso tratta i principi di programmazione e tutte le funzioni elementari del linguaggio di programmazione BASIC.

Esso si compone di 2 parti:

1. Un testo autodidattico diviso in 15 lezioni o "unità", ciascuna delle quali tratta un importante aspetto della programmazione.
2. 2 cassette di nastro con una raccolta di programmi che aiuteranno nello studio delle unità.

Va notato che questo corso insegna a scrivere utili e divertenti programmi per il computer ma non illustra tutti gli aspetti del linguaggio BASIC. Le caratteristiche più avanzate del BASIC sono spiegate a fondo e discusse nel secondo volume di questa serie.

INDICE

INTRODUZIONE

La COMMODORE porge il benvenuto agli utenti di questo corso di programmazione per il COMMODORE **16** e il **PLUS/4**. Questi due modelli forniscono un'ottima prestazione se utilizzati per eseguire giochi o per produrre suoni ed immagini su un normale apparecchio televisivo; ma essi sono anche, a tutti gli effetti, computer moderni e completi.

I computer sono apparecchiature estremamente versatili; in effetti la loro versatilità è inferiore solo a quella dell'essere umano. Il computer può ad esempio fungere da apparecchiatura didattica, da calcolatrice, da ausilio per gli handicappati, da elaboratore testi, da archivio elettronico per registrazioni finanziarie e controllo di magazzino, da monitor per i pazienti che si trovano in un'unità di cura intensiva, da apparecchio di controllo per i processi industriali o da computer ad uso scientifico utilizzabile per progettare edifici, centrali elettriche ed aeroplani.

I comandi ed i sistemi che essi controllano stanno acquisendo un'importanza sempre maggiore nelle attività quotidiane dell'uomo ed hanno già una vasta utilizzazione in molti dispositivi di pubblica utilità come, ad esempio, semafori, registratori di cassa e terminali bancari. Questa tendenza è destinata a svilupparsi ulteriormente in futuro. È in atto una autentica rivoluzione del computer a livello mondiale, che avrà delle conseguenze rilevanti quanto quelle causate a suo tempo dalla Rivoluzione Industriale.

La Rivoluzione del Computer non può essere arrestata, ma può essere influenzata, a livello personale, da ogni individuo che desideri farlo. Il mondo si sta suddividendo in due categorie di persone – i "passeggeri" e i "piloti". I "passeggeri" non controllano la situazione ma lasciano semplicemente che le cose accadano; possono apprezzare l'uso di prodotti basati sull'elaborazione dati o possono disprezzarlo, o entrambe le cose. Spesso rendono note le proprie opinioni, ma senza effetti rilevanti – in realtà non possono arrivare ad avere il controllo della situazione e, anche se lo potessero, non saprebbero come comportarsi.

Sono, al contrario, i "piloti" che controllano la rivoluzione. Essi inventano modelli di computer sempre più perfezionati e ne indirizzano l'applicazione agli scopi più disparati. I piloti hanno una pesante responsabilità poichè sono in grado di dare una svolta positiva alle tendenze attualmente in atto, cancellando così l'immagine di una società futura descritta spesso come disumana dalla fantascienza tradizionale.

Ma che cosa, in realtà, differenzia un "pilota" da un "passeggero"? Un'unica cosa: la comprensione del modo in cui funziona un computer. Ovviamente esistono vari livelli di comprensione. Molte persone se la cavano benissimo in una sala giochi anche se non sanno spiegare dal punto di vista tecnico il funzionamento dei video-games (in effetti, anche un gioco elettronico è controllato da un computer). Il livello di comprensione di cui si tratta in questa introduzione al Basic è molto più profondo, tale che si possa far fare al computer qualsiasi cosa si desideri, dal campo della didattica a quello delle applicazioni industriali o mediche, a un semplice gioco con cui trascorrere il tempo.

Per esercitare sul computer un controllo così totale, per renderlo una macchina servizievole, accurata e veloce nello svolgimento dei compiti assegnatigli, occorre essere in grado di programmarlo. La programmazione è la chiave per diventare "piloti".

Questo corso sarà completamente dedicato alla programmazione del **16** e del **PLUS/4**, ma una volta acquisita padronanza nella programmazione di queste apparecchiature, è molto semplice poi trasferire i concetti base anche ad altri computer, piccoli o grandi che siano.

Esercitarsi nell'applicazione delle norme relative alla programmazione è il modo migliore per acquistare dimestichezza con esse. La maggior parte delle persone è in grado di imparare agevolmente a programmare, una volta che gliene venga data la possibilità. Non occorrono conoscenze approfondite di matematica quanto, piuttosto, un luogo tranquillo dove poter leggere ed usare il computer, ed è consigliabile concedersi un certo tempo per completare il corso, senza accellerare inutilmente le varie fasi dell'apprendimento.

Il corso è suddiviso in quindici Lezioni. Ogni Lezione richiederà in media una o due serate di lavoro, applicandosi con una certa serietà. La maggior parte delle Lezioni comprende una parte teorica, una parte applicativa, alcuni esercizi di programmazione ed un questionario per accertarsi di avere ben compreso i concetti espressi nella Lezione. Ogni Lezione contiene degli esercizi che vanno spuntati mano a mano che si eseguono.

Quando la Lezione contiene delle domande, viene anche fornito uno spazio per le risposte. È consigliabile servirsi di questi spazi. Scrivere con una matita morbida, in modo che sia possibile cancellare le risposte nel caso in cui il corso debba essere successivamente utilizzato da altre persone. Se si utilizza una copia del corso già preventivamente usata, cancellare le risposte fornite dall'altro utente prima di iniziare

lo studio.

Nella programmazione i concetti sono strettamente legati gli uni agli altri. Gli argomenti appresi nelle Lezioni precedenti, sono poi menzionati ed utilizzati in seguito senza ulteriori spiegazioni. Per esempio, non è possibile comprendere il senso della Lezione 10 a meno che non si siano lette e comprese tutte le Unità da 1 a 9. Ciò rende necessario studiare le Lezioni nell'ordine in cui compaiono nel corso.

Al momento di iniziare lo studio di una nuova Lezione, è opportuno effettuare una prima lettura veloce della Lezione stessa dall'inizio alla fine. Anche se non si comprenderà tutto nei minimi dettagli, ciò consentirà almeno di farsi un'idea sull'argomento che ci si appresta a studiare.

Il passo successivo sarà quello di studiare dettagliatamente la Lezione. Ogni parte ha la sua importanza e a volte l'importanza delle singole parti è proporzionale alle difficoltà che si incontrano studiandole. Non trascurare nessun punto, ma cercare di capire bene tutto ciò che si incontra. Una volta appreso un concetto, è consigliabile ripeterlo mentalmente con parole proprie, per meglio assimilarlo. È anche probabile che alcune parti della Lezione vadano rilette diverse volte per essere pienamente comprese o che si debba tornare a consultare una Lezione già studiata in precedenza per chiarire i punti oscuri. È un fatto abbastanza normale quando si affronta un argomento tecnico.

Programmare è come suonare uno strumento musicale: si può imparare solo facendo pratica. È quindi opportuno svolgere tutti i problemi di programmazione del corso. Non appena è possibile, iniziare a creare e a risolvere problemi per conto proprio.

Una volta completato il corso si sarà in grado di usare il computer per molti scopi diversi – ad esempio fargli svolgere delle prove o dei quiz, fargli svolgere giochi di propria invenzione ed addirittura trovarlo utile per fare somme e tenere i conti. I giochi o altre applicazioni possono comprendere immagini colorate e create dall'utente nonchè suoni gradevoli e sgradevoli per sottolinearne il significato. –

La programmazione è in ogni caso un argomento molto vasto che non si può esaurire in un solo corso. Dopo un certo periodo di tempo, si vorrà probabilmente approfondirne i concetti. Si vorranno risolvere problemi più complicati oppure usare il computer come unità di controllo per un plastico ferroviario o per un centralino telefonico privato.

Il secondo volume di questa serie intitolato: Introduzione al BASIC Parte 2 sarà un ulteriore aiuto per approfondire queste conoscenze.

Ma ora, esauriti i preliminari, è tempo di affrontare la Lezione 1. Buona fortuna!

LEZIONE: 1

Questo corso è destinato in particolar modo ai possessori di due dei computer della gamma Commodore. Uno è il Commodore **16** e l'altro è il più sofisticato **PLUS/4**. Le due apparecchiature hanno contenitori di forma differente, capacità di memoria e strutture di tastiera diverse, ma sono fondamentalmente uguali per quanto riguarda i concetti fondamentali della programmazione. Lo stesso corso serve per entrambe.

Dove esistono differenze significative (questo si verifica principalmente nelle prime Lezioni del corso) le due apparecchiature vengono descritte separatamente. I capitoli vengono contraddistinti nel modo seguente:

Quanto segue riguarda il COMMODORE 16 (non si applica al Plus/4)

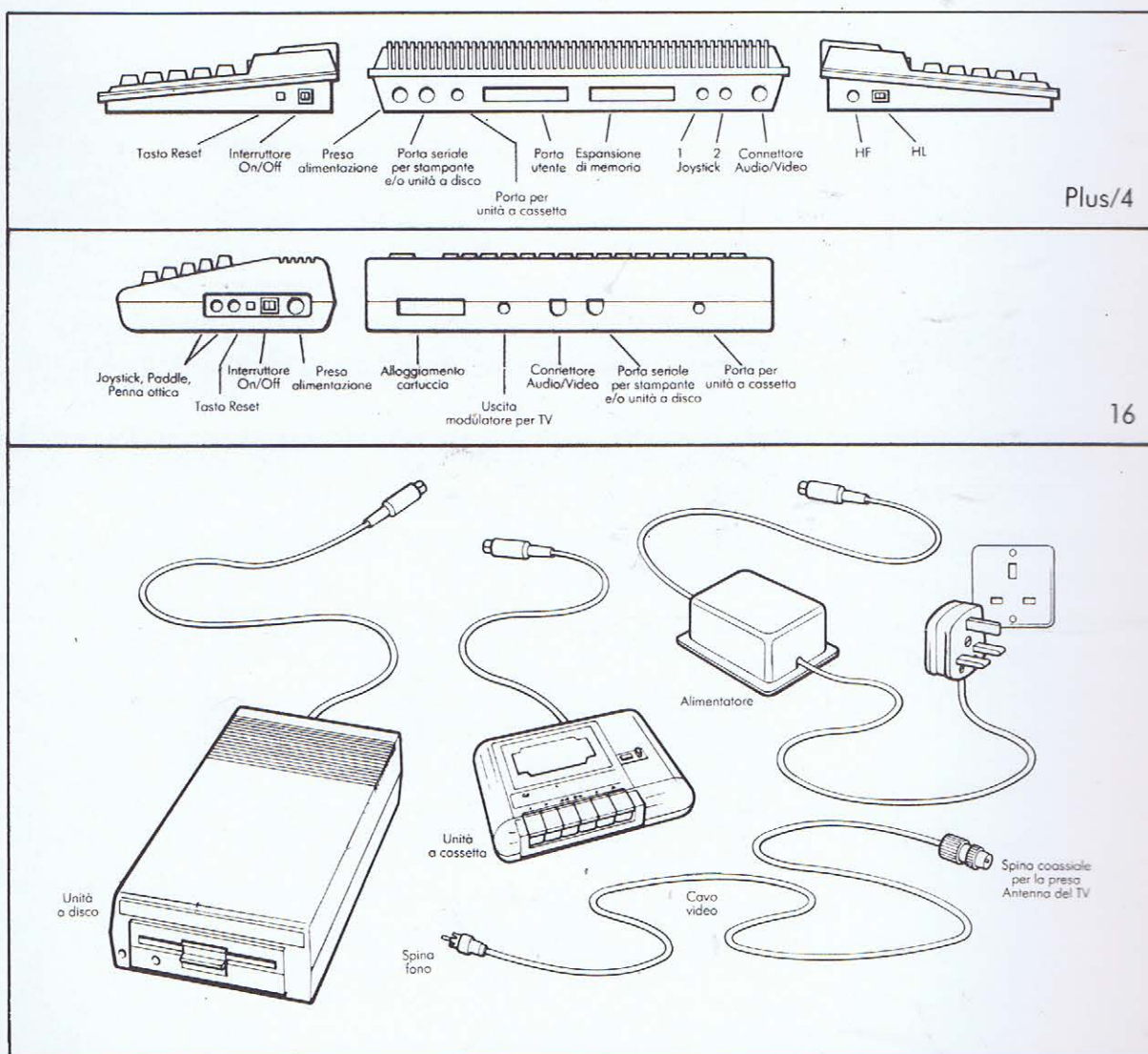
Quanto segue riguarda il Plus/4 (non si applica al Commodore 16)

La prima Lezione è destinata ai concetti fondamentali su come avviare il computer. Chiarifica un certo numero di punti che spesso sono difficili da comprendere e questioni pratiche che suscitano frequentemente seri problemi quando si acquista un computer per la prima volta.

L'apprendimento della programmazione richiede un ambiente adeguato. Dopo aver trovato un luogo che favorisca la concentrazione, applicarsi allo studio per un tempo non troppo breve (almeno 2 ore) e in un momento della giornata in cui la stanchezza non sia eccessiva. Accertarsi di aver preso tutte le precauzioni necessarie per evitare di essere disturbati; può essere, ad esempio, consigliabile appendere un cartello alla porta, staccare il telefono ed avvertire che non si desidera essere interrotti: è infatti molto difficile applicarsi allo studio della programmazione quando si è continuamente distratti.

Se si è installato ed utilizzato il computer, si può passare direttamente all'Esercizio 1. In caso contrario, è consigliabile dare almeno una lettura veloce alla Lezione, anche se si sa già di cosa si tratta.

Per prima cosa predisporre il computer collegandolo alla rete. Il computer ed il drive della cassetta o del disco vanno sistemati su un tavolo di fronte all'utente, l'alimentatore può essere posto sul pavimento, sotto il tavolo, e l'ap-

parecchio televisivo dovrebbe trovarsi ad una distanza di circa 2 metri, se si tratta di un apparecchio portatile. Apparecchi con schermi più grandi dovrebbero essere collocati ad una distanza maggiore. Le figure ed i testi visualizzati sono di dimensioni tali da poter essere letti anche da una certa distanza ed è comunque consigliabile non lavorare troppo vicini allo schermo.

Le diverse unità si collegano come mostrato nelle figure. Tutte le spine devono entrare nelle prese esercitando una pressione leggera ma decisa. Non cercare mai di forzare, ma prima di effettuare i collegamenti verificare attentamente la corrispondenza tra le spine e le prese.

I computer sono apparecchiature resistenti, ma le prese e le spine sono soggette a deterioramenti e guasti se collegate e disinserite troppe volte. Una volta che il computer è installato, è consigliabile accenderlo solo quando si ha intenzione di utilizzarlo.

Se si desidera utilizzare il televisore anche per la normale ricezione dei programmi, usare un commutatore di antenna che consente di mantenere collegati contemporaneamente il computer e l'antenna del televisore.

Sia il computer sia il televisore possono essere alimentati da un unico cavo di prolunga. È opportuno che il cavo sia piuttosto lungo, in modo da facilitare la sistemazione del computer. Se si usa un registratore a cassette, sono sufficienti due attacchi, ma se si possiede un drive per disco, è consigliabile utilizzare un adattatore a 4 vie. La presa di riserva potrà essere utile nel caso si decida di acquistare una stampante.

Se si decide di utilizzare una prolunga, è sconsigliabile effettuare personalmente i collegamenti, a meno che non si possieda una specifica esperienza a riguardo. Se questo consiglio ha fatto sorgere delle perplessità, è senz'altro meglio richiedere l'assistenza di un elettricista qualificato.

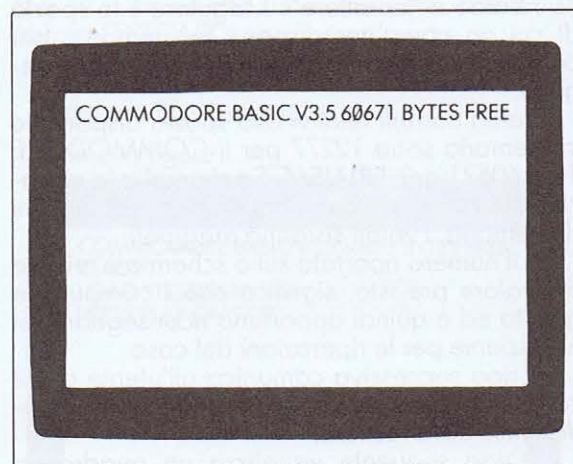
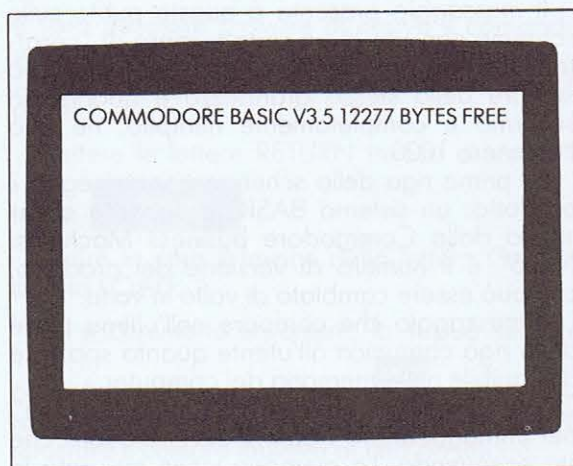
A questo punto si è pronti per l'accensione.

Accendere l'apparecchio televisivo. Se il televisore è fornito di una regolazione di sintonia a manopola, portarla sul "36". Se invece utilizza una selezione a pulsanti, selezionare un canale che non venga generalmente usato per la ricezione dei programmi.

Lo schermo del televisore apparirà vuoto e l'apparecchio emetterà probabilmente un forte sibilo. Se si desidera, abbassare il volume. Quindi accendere il computer, usando l'interruttore sulla destra. Se non ci sono inconvenienti, la spia rossa dovrebbe accendersi, ma generalmente a questo punto lo schermo televisivo è ancora vuoto.

Ora regolare la sintonia del televisore. La procedura di regolazione della sintonia varia con il modello del televisore e viene comunque illustrata nel libretto di istruzioni.

Nella maggior parte dei casi è presente una manopola o una piccola vite per ogni canale. Qualche volta i controlli della sintonia si trovano dietro un piccolo pannello. Se occorre utilizzare un cacciavite, evitare di introdurlo all'interno dell'apparecchio stesso, per evitare il rischio di subire una scossa elettrica.

Una volta regolata la sintonia, dovrebbe apparire la seguente immagine:

L'area centrale è bianca, con una cornice ciano (blu chiaro). Può darsi che per ottenere un'immagine stabile occorra agire sui comandi di controllo quadro. Se non appare l'immagine sopra descritta, spegnere il computer per alcuni secondi e poi provare di nuovo.

In caso di difficoltà, controllare i seguenti punti:

- L'apparecchio televisivo funziona? Verificare con la normale ricezione televisiva e se necessario farlo riparare.
- La spia di accensione del computer è accesa? Se non lo è controllare che:
  a) Non ci sia mancanza di corrente.
  b) Collegando alla presa che si sta utilizzando altre apparecchiature elettriche (ad esempio una lampada o un asciugacapelli), queste funzionino. Se non funzionano, provare a cambiare il fusibile interessato.
  c) Il fusibile nella presa dell'alimentazione del computer sia intatto (provare un nuovo fusibile).
  d) L'alimentatore sia collegato correttamente al computer.
- Il computer è collegato correttamente alla presa d'antenna del televisore?

Se il sistema ancora non funziona (un evento peraltro piuttosto improbabile), riportarlo al negoziante per chiarimenti ed eventuali riparazioni.

Il messaggio presente a questo punto sullo schermo consiste di un certo numero di "caratteri" (lettere e numeri). Questi caratteri sono sempre della stessa grandezza e quando lo schermo è completamente riempito, ne può contenere 1000.

La prima riga dello schermo contrassegna il prodotto: un sistema BASIC progettato e costruito dalla Commodore Business Machines. "V3.5" è il numero di versione del prodotto, che può essere cambiato di volta in volta.

Il messaggio che compare nell'ultima parte della riga comunica all'utente quanto spazio è disponibile nella memoria del computer.

Ogni computer necessita di una "memoria" per immagazzinare i dati relativi al lavoro che sta eseguendo. La memoria viene misurata in "byte", ognuno dei quali può contenere un "simbolo" o "carattere". Maggiore è lo spazio di cui un computer dispone in memoria, più complesso è il compito che è in grado di eseguire.

I valori corretti relativi allo spazio disponibile in memoria sono 12277 per il COMMODORE **16** e 60671 per il **PLUS/4**. Se si amplia la capacità del computer con l'aggiunta di espansioni di memoria, i valori saranno maggiori.

Se il numero riportato sullo schermo è minore del valore previsto, significa che il computer è guasto ed è quindi opportuno riconsegnarlo al negoziante per le riparazioni del caso.

La riga successiva comunica all'utente che il computer è ora pronto ad obbedire ai comandi battuti sulla tastiera.

La riga seguente visualizza un quadratino nero lampeggiante, chiamato cursore. Quando sulla tastiera si batte un comando, il cursore indica la posizione in cui verrà visualizzato il prossimo carattere. Provare per esempio a battere la seguente stringa:

PRINT5+8 RETURN

(che richiede la pressione di 9 tasti: vale a dire PRINT5+8 più il tasto RETURN, che si trova sulla destra della tastiera).

Prima di iniziare la battitura, accertarsi che il tasto SHIFT LOCK non sia bloccato. Mentre si batte ogni simbolo (ad eccezione di RETURN), questo appare sullo schermo ed il cursore si sposta in avanti di una posizione.

Il compito principale del tasto RETURN è di far sì che il computer esegua un'istruzione impartitagli: in questo caso si tratta di stampare (cioè di visualizzare) il risultato dell'operazione 5+8.

Per poter funzionare in modo adeguato, il computer deve contenere un programma.

I programmi vengono memorizzati su cassette o floppy disk e il primo Esercizio riguarderà il caricamento di un programma, da nastro o da disco, nel computer. Se si possiede un'unità a cassetta, eseguire le istruzioni immediatamente seguenti. Se invece si dispone di un drive per disco, passare direttamente all'Esercizio 1.2.

Caricamento di un programma da nastro:

1. Accertarsi che l'unità a cassetta sia collegata al computer.

1. Premere EJECT sull'unità a cassetta.

3. Aprire lo sportello del registratore ed inserire il primo dei due nastri allegati al corso, recante un'etichetta con la dicitura TAPE 1. Lo si può caricare con una qualsiasi delle due facciate rivolta verso l'alto, dal momento che sono perfettamente uguali. La parte esposta del nastro deve comunque essere rivolta verso l'utente. Chiudere lo sportello. Se non si chiude, non forzare ma accertarsi di aver introdotto il nastro correttamente.

4. Premere il tasto REWIND sul registratore. Controllare il nastro attraverso la finestrella e, se gira correttamente, attendere finchè non si arresta. Accertarsi di essere all'inizio del nastro.

5. Premere il tasto STOP sul registratore.

6. Battere ora il seguente messaggio:

LOAD"TESTCARD" RETURN

Ciò richiederà 15 battute di tasto in tutto, comprese le virgolette ("). Per ottenere il simbolo ", occorre tener premuto uno dei due tasti SHIFT, mentre si preme il tasto contrassegnato "/2.

Ricordare di rilasciare il tasto SHIFT appena il segno " appare sullo schermo (ma non prima).

A questo punto si dovrebbe ottenere il messaggio appropriato. Alcuni degli errori più comuni da evitare sono:

Battere tenendo premuto il tasto SHIFT LOCK. Apparirà un disegno con linee, cuori e picche e non accadrà nulla di significativo.

Usare due volte il segno di apice (') anzichè una volta il carattere virgolette ("). Il computer visualizzerà il messaggio:

?SINTAX ERROR
READY

e sarà possibile provare di nuovo il comando sulla riga successiva.

Lasciare uno spazio tra " e T, TEST e CARD, o D e ".

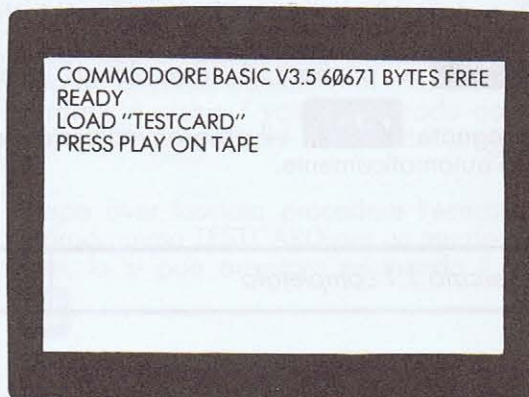
Battere le lettere RETURN invece di usare il tasto RETURN.

Usare la cifra Ø invece della lettera O nella parola LOAD.

Se si commette un errore, lo si può sempre cancellare utilizzando il tasto INST DEL. Ogni battuta di questo tasto cancella un carattere e fa spostare il cursore indietro di una posizione.

7. Se si fornisce un messaggio corretto (o anche se si commette un errore nello scrivere la parola TESTCARD) verrà visualizzato:

PRESS PLAY ON TAPE
(premere PLAY sul registratore)

Come in figura:

Premere il tasto PLAY sull'unità a cassetta. Il contenuto dello schermo scomparirà e il nastro inizierà a girare. Dopo circa trenta secondi apparirà il messaggio:

FOUND TESTCARD

e il nastro si arresterà. Ciò significa che l'avvolgimento del nastro ha raggiunto l'inizio della sezione sulla quale è memorizzato il programma TESTCARD. Per caricare rapidamente il programma, premere sulla tastiera un qualsiasi tasto relativo ad una lettera (in effetti se non si esegue alcuna operazione per un tempo abbastanza lungo, il programma verrà caricato automaticamente). Quando si è premuto il tasto, il contenuto dello schermo verrà cancellato di nuovo e il trasferimento del programma dal nastro al computer richiederà circa tre minuti.

Se, dopo questo tempo, il computer visualizza il messaggio

FOUND HANGMAN

significa che il nome TESTCARD è stato scritto in maniera errata al momento di battere il mes-

saggio originale. Arrestare il computer premendo il tasto RUN STOP e tornare al punto 4.

Nella maggior parte dei casi questa procedura funziona perfettamente. Se ciò non avviene e il nastro continua a girare senza che accada nulla, è possibile che il nastro stesso sia rimasto in qualche modo danneggiato. Provare a girarlo e a caricare il programma dall'altra facciata. Se ancora non si riesce a caricare il programma TESTCARD, riportare il nastro, l'unità a cassetta ed il computer al proprio negoziante per un controllo accurato. Ciò non è quasi mai necessario, dal momento che tutti i computer Commodore utilizzano un particolare metodo di registrazione che è molto più affidabile di quello utilizzato dalla maggior parte delle altre marche.

Quando il programma TESTCARD è caricato, verrà visualizzato

READY.

Avviare il programma battendo:

RUN RETURN

(4 battute di tasto)

Si può, in alternativa tenere premuto il tasto SHIFT e premere il tasto lungo contrassegnato f3 f6 : il programma verrà avviato automaticamente.

Esercizio 1.1 completato	

ESERCIZIO 1·2

(Per gli utenti in possesso di unità a floppy disk).

Importante: Leggere fino in fondo le seguenti istruzioni, prima di provare a metterle in pratica.

1. Assicurarsi che il drive sia collegato sia alla presa di corrente sia al computer. Sulla parte posteriore dell'unità a disco 1541 si trovano due prese identiche per il collegamento ed è possibile usare una qualsiasi delle due. Per il momento non cercare di inserire un disco nell'unità.

2. Accendere il drive, utilizzando l'interruttore posto sul retro. Sia la spia verde sia quella rossa, poste sulla parte anteriore, si accenderanno. Per circa un secondo, il drive eseguirà una autodiagnosi per accertarsi del proprio corretto funzionamento. Se non ci sono inconvenienti, la spia rossa si spegne, e solo quella verde rimane accesa. Se la spia rossa non si spegne, o inizia a lampeggiare, è probabile la presenza di inconvenienti sul drive. Provare a spegnere ed accendere di nuovo e se l'inconveniente continua a verificarsi, riportare il drive al proprio negoziante per le riparazioni del caso.

3. Aprire il drive spingendo la leva verso l'interno e in alto. Spingere il disco programma delicatamente a fondo nella fessura. L'etichetta recante la dicitura COMMODORE deve trovarsi in alto, verso l'utente e sulla destra.

Premere la leva verso il basso fino a che non

scatti in avanti. Il disco è ora caricato.

4. Localizzare il tasto lungo contrassegnato f2 f5 e premerlo. Quindi battere la parola

TESTCARD

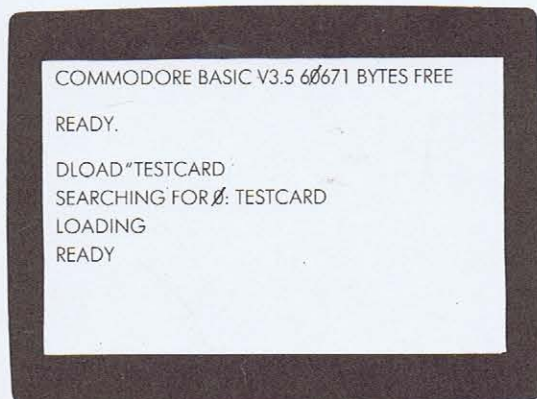
Sullo schermo dovrebbe apparire il messaggio:

DLOAD"TESTCARD

Se ciò non si verifica, si possono cancellare i caratteri errati premendo il tasto INST DEL. Ogni battuta di questo tasto cancella un carattere e fa spostare il cursore all'indietro di una posizione.

Quando si è sicuri che il messaggio battuto sia corretto, premere il tasto RETURN.

Se si batte il messaggio correttamente, il drive emetterà un leggero ronzio, mentre la spia rossa si accenderà. Infine la spia si spegnerà e sullo schermo apparirà il messaggio:

```
COMMODORE BASIC V3.5 6Ø671 BYTES FREE
READY.
DLOAD"TESTCARD
SEARCHING FOR Ø: TESTCARD
LOADING
READY
```

Se l'operazione non avviene nel modo indicato, le cause probabili possono essere due:

a) Se il drive non è stato collegato correttamente o non è stato acceso, apparirà il messaggio:

DEVICE NOT PRESENT

b) Se il disco non è stato caricato, o non è stato caricato in maniera corretta, oppure se è stato caricato il disco sbagliato, apparirà il messaggio:

FILE NON FOUND

In entrambi i casi identificare la causa, correggerla e provare di nuovo.

Una volta che il programma è stato caricato correttamente, può essere avviato battendo:

RUN RETURN
(4 battute di tasto)

Un altro modo per lanciare il programma è quello di premere il tasto SHIFT tenendo premuto il tasto f3 f6.

Prima di procedere si consiglia di prestare attenzione ad alcune precauzioni da prendere riguardo alla manutenzione dei dischi.

- Non spegnere o accendere il drive mentre un disco si trova al suo interno. Eseguire il caricamento sempre dopo l'accensione, e rimuovere il disco prima dello spegnimento del drive.
- Maneggiare il disco il meno possibile; proteggerlo da polvere, sbalzi di temperatura e umidità e non toccare mai la superficie esposta in corrispondenza della fessura della copertina.
- Mantenere i dischi lontani da apparecchiature elettriche come, ad esempio, televisori, motori elettrici e rilevatori di metalli. Se si porta con sè un disco attraverso il posto di controllo di un aeroporto, non lasciare che venga esposto ai raggi X, ma piuttosto, insistere per essere perquisiti manualmente.

(Coloro che utilizzano come memoria magnetica un normale registratore a cassette possono riprendere la lettura da questo punto).

Il primo programma mostra un esempio delle varie possibilità di creazione di colori e suoni che il computer può produrre. Consente anche la regolazione dell'apparecchio televisivo, in modo da ottenere sfumature di colore ottimali. Ricordare di alzare il volume in modo da rendere udibili le emissioni sonore del programma.

Dopo aver lasciato procedere l'esecuzione del programma TESTCARD per un tempo sufficiente, la si può arrestare premendo il tasto RUN STOP.

Quando di preme questo tasto (o qualora il programma, per una qualsiasi ragione, dovesse arrestarsi) sullo schermo appare il seguente messaggio:

BREAK IN 56Ø

READY.

Il numero 56Ø dell'esempio può essere in realtà qualsiasi altro numero. BREAK non significa che il computer sia guasto: semplicemente comunica all'utente che si è verificata un'interruzione nella sequenza dei comandi che costituiscono il programma.

Esercizio 1.2 completato	

ESERCIZIO 1.3

L'esercizio 1.3 è costituito da un gioco ad indovinelli, che facilita l'apprendimento dell'uso della tastiera. Il programma è chiamato HANGMAN. Può essere caricato o da nastro o da disco, utilizzando i comandi

LOAD"HANGMAN" RETURN

per il nastro o

DLOAD"HANGMAN" RETURN

per il disco. Ricordare che il tasto f2 f5 fornisce automaticamente la sequenza:

DLOAD"

Quando il programma è caricato ed appare il messaggio READY, battere:

RUN RETURN

o premere il tasto f3 f6 mentre si tiene premuto il tasto SHIFT e verrà dato inizio al gioco. Se si ignorano le regole, è sufficiente battere alcune lettere ed osservare il risultato (visivo e acustico). Questa semplice operazione consentirà all'utente di comprendere in breve tempo il funzionamento del gioco.

Giocare per tutto il tempo che si desidera, e approfittare di questo piacevole passatempo per acquisire dimestichezza nell'uso delle lettere della tastiera.

Esercizio 1.3 completato	

LEZIONE: 2

Questa Lezione è dedicata alla tastiera e permette di imparare ad usarla per scrivere messaggi ed eseguire disegni sullo schermo.

Se si è già usata una normale macchina per scrivere, la tastiera del computer avrà un aspetto familiare. La posizione delle lettere, dei numeri e della maggior parte dei simboli è uguale a quella della macchina per scrivere e sono presenti anche i soliti tasti SHIFT e SHIFT LOCK - sebbene in un computer abbiano delle funzioni leggermente diverse.

D'altra parte, anche se non si è mai usata una macchina per scrivere, è possibile imparare ad usare la tastiera del computer. Ciò richiederà probabilmente più tempo, ma non si troveranno difficoltà particolari.

Solo nel corso di questa Lezione, si prega di non usare il tasto RETURN a meno che non sia richiesto dal testo. Come si è visto nella Lezione 1, questo tasto fa sì che il computer esegua effettivamente un comando, come ad esempio caricare un programma o sommare dei numeri. In questo momento, per utilizzare lo schermo, non occorre l'aiuto del computer.

Se si preme il tasto RETURN il computer cercherà di obbedire al messaggio o alla figura che si è appena battuta, interpretandolo erroneamente e deformandone la visualizzazione.

Un altro simbolo da evitare in questo momento sono le virgolette ("). Questo segno ha un significato particolare ed altera il modo in cui lo schermo reagisce alla pressione di molti degli altri tasti. Se sullo schermo compare un segno di virgolette, può essere molto più difficile l'esecuzione di disegni. In una Lezione successiva, verrà illustrato il significato di questo simbolo, ma, per ora, si raccomanda di non usarlo.

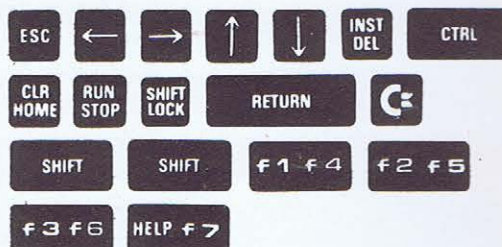
Questa lista di "cose da non fare" può apparire allarmante. Ecco un'altra "cosa da non fare": preoccuparsi!

A differenza di quanto avviene nei film di fantascienza, i computer Commodore non possiedono un comando di "auto-distruzione". È assolutamente impossibile danneggiare il computer battendo sulla tastiera. Alcune combinazioni di caratteri contenenti " o RETURN possono, casomai, provocare delle reazioni apparentemente incomprensibili da parte dell'apparecchiatura ed alcune sequenze, che potrebbero essere battute inavvertitamente, faranno sì che il computer cessi di reagire ai comandi. Questi inconvenienti sono solo temporanei: si possono facilmente risolvere spegnendo ed accendendo di nuovo il computer o premendo il tasto RESET a fianco dell'interruttore principale.

Il COMMODORE **16** ha 66 tasti sulla tastiera ed il **PLUS/4** ne ha 67. Entrambe le tastiere vengono illustrate qui di seguito.

I tasti si dividono in due categorie:

- 48 tasti 'simbolo' che fanno sì che il computer visualizzi dei caratteri sullo schermo. (La barra spaziatrice viene considerata un tasto simbolo).
- 18 (o 19) tasti 'funzione', che controllano il modo in cui vengono visualizzati i caratteri.

Nel COMMODORE **16**, i tasti funzione sono:

Nel COMMODORE **PLUS/4**, i tasti funzione sono:

COMMODORE **16**

COMMODORE **PLUS/4**

L'unica differenza fra le due tastiere (oltre alla diversa disposizione dei tasti) è che nel **PLUS/4** il tasto Controllo è doppio.

Premere il tasto SHIFT LOCK più volte e notare che può assumere due posizioni - sollevata ed abbassata. Infine, assicurarsi che sia sollevato.

Procedere ora ad un avviamento normale o premere il tasto RESET se il computer è già acceso. Immediatamente al di sotto del messaggio READY comparirà il cursore lampeggiante.

In questo Esercizio, verrà preso in esame il modo in cui il cursore si sposta quando sullo schermo vengono visualizzati dei simboli. Lo scopo del cursore è quello di indicare la posizione in cui verrà visualizzato il prossimo carattere battuto. Battere alcune lettere ed osservare il cursore. Notare che ogni carattere sostituisce il cursore, che quindi si sposta nella posizione successiva.

A questo punto riempire di lettere l'intera riga, finché il cursore non si trovi all'estremità dell'area bianca. Battere ancora una lettera ed osservare ciò che accade: il cursore si sposta automaticamente all'inizio della riga successiva.

Prima di procedere, contare il numero di lettere presenti sullo schermo e riempire il quadratino vuoto qui di seguito:

Ci sono [ ] spazi per i caratteri su ogni riga dello schermo.

Quindi battere alcune altre righe, finché non si giunge in fondo allo schermo. Per facilitare questa operazione, si può tenere premuto qualsiasi tasto relativo ad un simbolo; il simbolo verrà automaticamente ripetuto e riempirà velocemente lo schermo. Contare il numero delle righe visualizzate e scriverlo nel quadratino bianco qui di seguito. Ricordare di includere nel numero le righe vuote al di sopra e al disotto del messaggio iniziale.

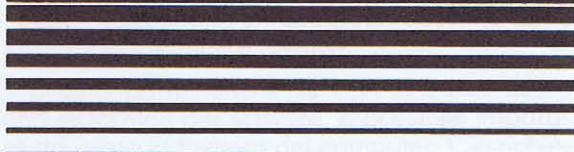
COMMODORE BASIC...

In uno schermo possono essere contenute [ ] righe di caratteri.

Riempire a questo punto l'ultima riga, finché il cursore raggiunge l'angolo in basso a destra dello schermo. Battere ancora un carattere ed osservare il cursore. L'intero schermo si sposta verso l'alto ed il cursore si sposta all'inizio della riga vuota successiva, che compare ora in fondo allo schermo. Ogni nuova riga vuota che appare provoca la sparizione della prima riga in alto. La riga in alto è sparita definitivamente e non c'è modo di farla riapparire a meno che non si sia precedentemente memorizzata.

Riempire alcune altre righe per constatare che il sistema continua a fornire ulteriore spazio in fondo allo schermo per l'aggiunta del testo.

Esercizio 2.1 completato	

ESERCIZIO 2·2

(Per il COMMODORE **16**)

Per i simboli sono previsti 48 tasti, ma può essere visualizzato un numero molto maggiore di simboli diversi, comprendenti lettere, numeri, simboli matematici e di punteggiatura e un'ampia gamma di 'simboli grafici' o forme semplici che possono essere combinate per creare immagini. Tutti questi caratteri possono essere selezionati usando uno dei due tasti SHIFT (collegati tra di loro all'interno del computer) e il tasto speciale 'Commodore' contrassegnato C=

Reinizializzare la macchina e battere la sequenza

1234567890QWERTYUIOP@+–
ASDFGHJKL:;★ZXCVB

(In questo modo sono stati battuti, fila per fila, tutti i tasti relativi ai simboli).

Tenere ora premuto uno dei tasti SHIFT e battere di nuovo la riga. Si otterrà in questo modo una riga di simboli quasi completamente diversa (compreso il carattere ", ma ciò non causerà alcun inconveniente se si seguono le istruzioni). Copiare i simboli nella seconda riga della tabella qui di seguito e notare che alcuni dei simboli grafici (ad esempio quelli corrispondenti ai tasti U e I) combaciano. Notare anche che i simboli grafici solitamente toccano il margine dei piccoli quadrati che occupano, in modo da potersi unire gli uni con gli altri.

Battere quindi la riga una terza volta, tenendo premuto il tasto C= .Molti segni cambieranno nuovamente. Copiare i simboli nella terza riga della tabella.

Per esaminare gli altri simboli grafici, ripetere l'Esercizio con la sequenza NM,./£=, facendola seguire da 33 spazi e giungendo così all'estremità dello schermo.

È importante notare e ricordare che la cifra Ø è diversa dalla lettera O. È opportuno utilizzare sempre il segno Ø (barrato) per indicare lo zero, differenziandolo così a prima vista dalla lettera O.

Premere C= e SHIFT contemporaneamente. Le lettere maiuscole sullo schermo diventeranno minuscole, mentre alcuni dei simboli grafici diventeranno lettere maiuscole. Premere di nuovo i tasti contemporaneamente (o continuare a tenerli premuti) e riappariranno le lettere maiuscole. Viene qui anticipato il concetto generale, che è possibile utilizzare o una serie completa di simboli grafici o una serie limitata di essi e le lettere maiuscole, ma non entrambe le cose contemporaneamente. L'uso delle lettere minuscole verrà illustrato nel secondo volume di questo corso.

SIMBOLO	1	2	3	4	5	6	7	8	9	Ø	Q	W	E	R	T	Y	U	I	O	P	@	+	–	A	S	D	F	G	H	J	K	L	:	;	★	Z	X	C	V	B
SHIFT																																								
C=																																								

SIMBOLO	N	M	,	.	/	£	=
SHIFT							
C=							

(Per il COMMODORE **PLUS/4**)

Anche in questo caso sono previsti 48 tasti per i simboli, ma può essere visualizzato un numero molto maggiore di simboli diversi, comprendenti lettere, numeri, simboli matematici e di punteggiatura e un'ampia gamma di 'simboli grafici' o forme semplici che possono essere combinate per creare immagini. Tutti questi caratteri possono essere selezionati usando uno dei due tasti SHIFT (collegati tra di loro all'interno del computer) e il tasto speciale 'Commodore' contrassegnato C= .

Riavviare il computer (o premere il tasto RESET) e battere la sequenza

1234567890QWERTYUIOP@+–
ASDFGHJKL:;★ZXCVB

(In questo modo sono stati battuti, fila per fila, tutti i tasti relativi ai simboli).

Tenere ora premuto uno dei tasti SHIFT e battere di nuovo la riga. Si otterrà in questo modo una riga di simboli quasi completamente diversa (compreso il carattere ", ma ciò non causerà alcun inconveniente se si seguono le istruzioni). Copiare i simboli nella seconda riga della tabella riportata qui di seguito e notare che alcuni dei simboli grafici (ad esempio quelli corrispondenti ai tasti U e I) combaciano. Notare anche che i simboli grafici solitamente toccano il margine dei piccoli quadrati che occupano, in modo da potersi unire gli uni con gli altri.

Battere quindi la riga una terza volta, tenendo premuto il tasto C= .Molti simboli cambieranno nuovamente. Copiare i simboli nella terza riga della tabella.

Per esaminare gli altri simboli grafici, ripetere l'Esercizio con la sequenza VBNM,./£=, facendola seguire da 31 spazi e giungendo così all'estremità dello schermo.

È importante notare e ricordare che la cifra 0 è diversa dalla lettera O. È opportuno utilizzare sempre il segno 0 (barrato) per indicare lo zero, differenziandolo così a prima vista dalla lettera O.

Premere C= e SHIFT contemporaneamente. Le lettere maiuscole sullo schermo diventeranno minuscole, mentre alcuni dei simboli grafici diventeranno lettere maiuscole. Premere di nuovo tasti contemporaneamente (o continuare a tenerli premuti) e riappariranno le lettere maiuscole. Viene qui anticipato il concetto generale, che è possibile utilizzare o una serie completa di simboli grafici o una serie limitata di essi e le lettere maiuscole, ma non entrambe le cose contemporaneamente. L'uso delle lettere minuscole verrà illustrato nel secondo volume di questo corso.

SIMBOLO	1	2	3	4	5	6	7	8	9	0	+	–	=	Q	W	E	R	T	Y	U	I	O	P	@	£	★	A	S	D	F	G	H	J	K	L	:	;	Z	X	C
SHIFT																																								
C=																																								

SIMBOLO	V	B	N	M	,	.	/																															
SHIFT																																						
C=																																						

Esercizio 2.2 completato	

ESERCIZIO 2·3

Finora ci si è limitati ad utilizzare caratteri in sequenza, da sinistra a destra e dall'alto in basso.

Questo costituisce un modo piuttosto limitato per eseguire un disegno, ed è molto più conveniente sfruttare la possibilità di disporre testi e simboli grafici in qualsiasi posizione desiderata.

Ciò può essere ottenuto con i tasti cursore, che sono cinque:

CLR HOME, ←, →, ↑ e ↓. Nel **PLUS/4** (e non nel **16**) i tasti freccia hanno effettivamente la forma di una freccia.

Quando si batte il tasto CLR HOME da solo, si provoca lo spostamento del cursore all'indietro nella posizione 'Home', che è l'angolo in alto a sinistra dello schermo. Reinizializzare il computer o premere questo tasto. Il cursore si sposterà nell'angolo dello schermo.

I quattro tasti freccia vengono usati per spostare il cursore nella direzione della freccia. Se ora si preme ↓, il cursore si sposta di una riga verso il basso, così da trovarsi al di sopra della C di COMMODORE. La C rimane visibile poichè il cursore è trasparente. Provare a spostare il cursore lungo la riga, usando il tasto →, e quindi spostarlo di nuovo nella posizione iniziale con ←. Nessuno dei caratteri che si trovano nella riga verrà modificato finchè si utilizzano i soli tasti cursore.

Se si preme un tasto simbolo, il carattere al di sotto del cursore viene sostituito dal nuovo carattere ed il cursore avanza di una posizione. Per esempio, se si pone il cursore sulla C di COMMODORE e si battono le lettere MAGNIFICO, sulla riga apparirà

MAGNIFICO BASIC...

(Naturalmente, qualsiasi altra parola di nove lettere potrà servire ugualmente a questo esempio).

Utilizzare ora il tasto → per spostare il cursore alla B di BYTES e modificare questa parola in GATTO.

Come tutti gli altri tasti, i tasti controllo cursore hanno un effetto ripetuto, cioè tenendo premuto uno di questi tasti, il cursore continuerà a muoversi di circa 10 posizioni di carattere al secondo. Ciò è utile per gli spostamenti rapidi lungo lo schermo.

Se si preme il tasto → quando il cursore si trova alla fine di una riga, questo si sposta all'inizio della riga successiva. In maniera analoga, la pressione del tasto ← all'inizio di una riga fa spostare il cursore alla fine della riga precedente.

I tasti ↑ e ↓ fanno spostare il cursore direttamente verso l'alto e verso il basso, senza bisogno di movimenti laterali.

Le due estremità superiore ed inferiore dello schermo presentano delle caratteristiche particolari, che è bene prendere in esame.

Riempire lo schermo di alcune righe di caratteri e spostare il cursore alla fine dell'ultima riga. Quando il cursore viene spostato oltre la fine di questa riga, utilizzando → o ↓, l'intero schermo si sposta verso l'alto, proprio come se si fosse aggiunto un altro carattere. La riga superiore scompare.

Tornare ora nella posizione 'home' e provare a spostare il cursore all'indietro. Lo schermo non si sposta verso il basso come ci si potrebbe aspettare; non accade nulla ed il cursore rimane nella stessa posizione.

A questo punto premere CLR HOME mentre si tiene premuto uno dei tasti SHIFT. Il cursore si sposta nella posizione "home" e il contenuto dello schermo scompare, fornendo quindi all'utente uno schermo vuoto su cui è possibile lavorare.

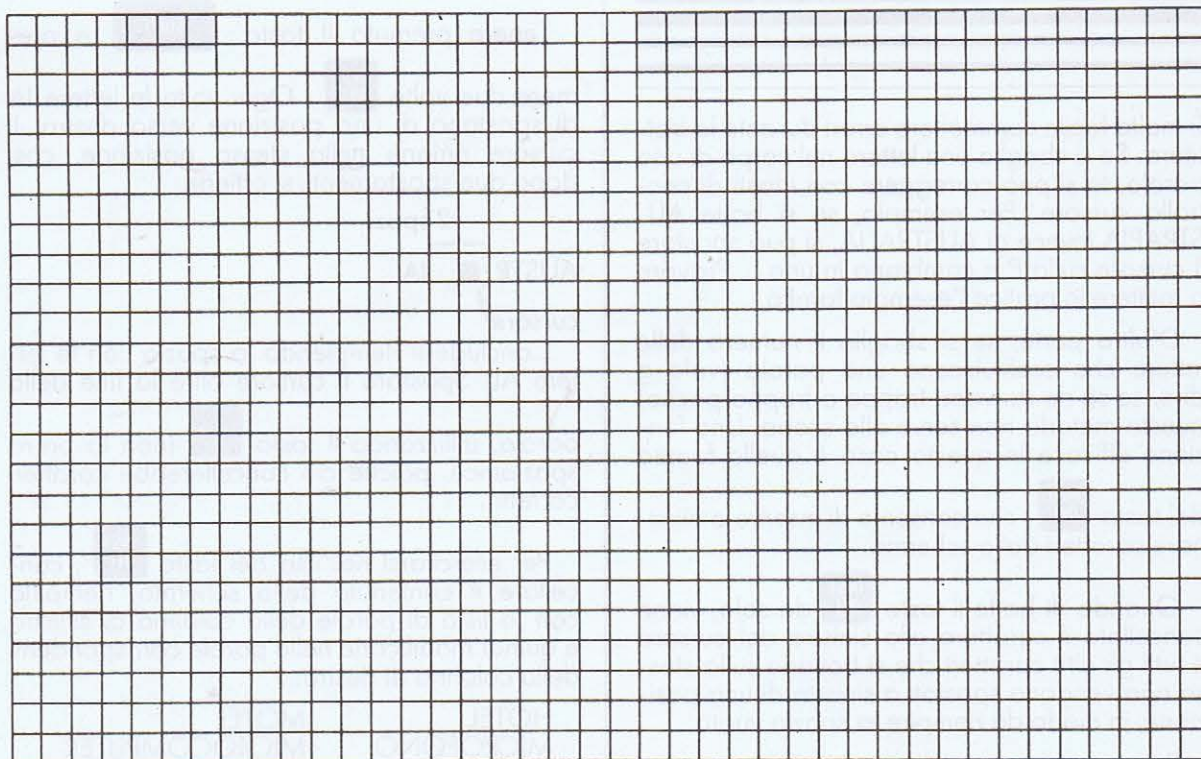
Esercitarsi nell'eseguire disegni sullo schermo utilizzando i simboli grafici ed i tasti controllo cursore. Iniziare con alcune figure geometriche elementari come quadrati, rettangoli, triangoli e piccoli cerchi. Se si commette un errore, spostare il cursore indietro e battere il carattere corretto. L'uso della barra spaziatrice eliminerà caratteri che si trovino in posizioni erronee.

Quando si è acquisita una certa dimestichezza nell'uso dei simboli grafici, disegnare un riquadro come quello che segue contenente il proprio nome:

MARIO
ROSSI

Disegnare ora alcune carte da gioco, con gli angoli arrotondati ed i simboli corretti (è consigliabile limitarsi a carte con semi neri e con valori fino a 1Ø).

Infine, se si è dotati di un certo talento artistico, provare a disegnare qualcosa come un animale, una astronave o un volto umano.

Per prima cosa pianificare l'immagine che si intende visualizzare, utilizzando la griglia riportata qui di seguito:

	Esercizio 2.3 completato

ESERCIZIO 2.4

È molto facile commettere errori durante la battitura. Se si sbaglia una lettera nel corpo di una parola, la si può correggere con i tasti di controllo cursore. Per esempio, se si batte AUSTRAPIA invece di AUSTRALIA, si può spostare il cursore sulla P e cambiarla in una L. Provare a mettere in pratica l'esempio fornito.

D'altra parte, se si sbaglia il numero delle lettere che costituiscono una parola (vale a dire, se se ne scrivono troppe o troppo poche) questo metodo non serve allo scopo. Una funzione efficace in questo caso è quella fornita dal tasto INST DEL, che consente di inserire o eliminare caratteri dallo schermo.

Quando di batte il tasto INST DEL da solo, viene cancellato il carattere alla sinistra del cursore e tutti gli altri caratteri che si trovano sulla stessa riga vengono spostati a sinistra di una posizione, in modo da riempire lo spazio vuoto.

Per esempio, supponiamo che si sia erroneamente battuto INXDIA invece di INDIA. Occorre eliminare la X. Collocare il cursore al di sopra della D (e non della X) e premere INST DEL. La X scompare e DIA si sposta di una posizione, formando la parola INDIA senza nessuno spazio nel mezzo.

Ora provare ad usare il tasto INST DEL per eseguire altre correzioni, come ad esempio:

CINA da CAINA
EGITTO da EEGITTO
FINLANDIA da FINLLANDIA
AUSTRIA da AUSTRALIA

In pratica, l'uso più comune del tasto è l'eliminazione di caratteri che si sono battuti. L'uso di questo tasto, in un unico movimento, elimina l'ultimo simbolo e riposiziona il cursore. Ben presto ci si abituerà a premere INST DEL ogni qualvolta si commetteranno degli errori di battitura.

L'altra funzione del tasto INST DEL può essere ottenuta premendolo insieme al tasto SHIFT. Questa funzione viene usata per inserire spazi nel mezzo di parole o righe. Questi spazi possono essere riempiti con caratteri nel modo usuale.

Provare l'esempio seguente, che consiste nel modificare in AUSTRALIA la parola AUSTRIA.

Cancellare lo schermo (SHIFT e CLR HOME) e battere

AUSTRIA

Spostare il cursore all'indietro sulla I.

Tenere premuto il tasto SHIFT e premere due volte INST DEL. Ogni volta le lettere IA di spostano di una posizione verso destra. Il cursore rimane nella stessa posizione, così dopo due spostamenti, si ottiene

```
      2 spazi
AUSTR ■  IA
     ↑
cursore
```

Concludere riempiendo lo spazio con le lettere AL. Spostare il cursore oltre la fine della parola, utilizzando il tasto → (non la barra spaziatrice, poiché ciò cancellerebbe caratteri corretti).

Per esercitarsi nell'uso del tasto INST DEL, cancellare il contenuto dello schermo, riempirlo con la lista di parole della colonna di sinistra e quindi modificarle nelle parole corrispondenti della colonna di destra:

HOTEL	MOTEL
MICROFONO	MICROCOMPUTER
BOSCAGLIA	BOSCO
ANGOLO	ANGELO
CAPPA	CAPITANO
SCRITTORE	SCRITTURA
ATTORE	AUTORE
PALLA	PALLONE
FILO	FILANDA
FIORITO	FIORE
LUNA	LUNOTTO
PICCHIARE	PICCIONE
TATTO	TACITO
CAVALLO	CAVALLERIA
FINALE	FORTUNALE
TAXI	TASSA
ROSSO	ROSSORE
MARIO	MERCURIO
GIOVE	GIOVIALE
PAL	PASCAL
BACCO	BASIC
GIAVA	GIAMAICA

A questo punto riportare tutte le parole nella forma originale.

Esercizio 2.4 completato	

Il programma della Lezione 2 è denominato SPEEDTYPE ed è uno strumento utile per acquisire familiarità con la tastiera.
Caricarlo battendo

LOAD"SPEEDTYPE"

o

DLOAD"SPEEDTYPE

Avviare il programma con il comando RUN ed esercitarsi al suo uso per tutto il tempo che si ritenga necessario.

LEZIONE: 3

Sia il COMMODORE **16** sia il **PLUS/4** sono computer a colori. Questa Lezione fornisce un'introduzione all'uso di queste apparecchiature per eseguire disegni colorati sullo schermo del televisore.

Se l'apparecchio televisivo di cui si dispone è un modello in bianco e nero, eseguire le stesse istruzioni, ma ovviamente, le immagini appariranno in bianco e nero.

ESERCIZIO 3·1

Utilizzare il programma TESTCARD (Lezione 1) per assicurarsi che l'apparecchio televisivo sia correttamente regolato.

Arrestare il programma premendo il tasto .

Si noterà che a questo punto il cursore è nero. Ora il cursore può cambiare colore e, oltre a fornire la posizione in cui verrà visualizzato il simbolo successivo, indicherà in tal modo anche il colore che il simbolo assumerà sullo schermo. Provare a battere una sequenza di simboli (tasti e I) e notare che appaiono in nero, cioè nel colore corrente del cursore.

Il colore del cursore può essere modificato in qualsiasi momento battendo uno degli otto tasti colore, mentre si tiene premuto il tasto CTRL. I tasti colore sono contrassegnati con numeri dall'1 a 8 e riportano anche le abbreviazioni dei colori a cui corrispondono.

Tenere premuto CTRL e battere in successione tutti i tasti colore.

Quando si preme 1 (BLK) il colore del cursore diventa nero (a meno che non lo sia già).

Quando si preme 2 (WHT) il colore del cursore diventa bianco.

Gli altri tasti modificano il colore del cursore rispettivamente in rosso, viola, porpora, verde, blu e giallo. Quando il cursore è bianco, è invisibile contro lo sfondo bianco: sembra che sia scomparso.

Per ottenere una gamma di altri otto colori, utilizzare il tasto invece di CTRL.

Si tratta delle tonalità pastello meno brillanti:

e 1 arancio

e 2 marrone

C= e 3 giallo-verde

C= e 4 rosa

C= e 5 verde grigio

C= e 6 blu chiaro

C= e 7 blu scuro

C= e 8 verde chiaro

Ora provare ad eseguire dei disegni colorati. È consigliabile iniziare con delle barre colorate di lunghezza variabile. Usare il simbolo grafico ▭ (C= e U) per tracciare ogni barra.
Per esempio, per creare una barra rossa con una lunghezza corrispondente a 5 simboli, per prima cosa tenere premuto CTRL e battere 3 (ROSSO); ciò modificherà in rosso il colore del cursore. Quindi tenere premuto C= e battere cinque volte U.

È possibile cambiare il colore del cursore in qualsiasi momento si desideri. Quando si è acquisita una certa familiarità con i tasti colore, provare a tracciare un "diagramma dei risultati elettorali", come mostrato nella figura 1. Mantenere nero il colore delle lettere per evidenziare il colore delle barre.

Figura 1

Esercizio 3.1 completato	

ESERCIZIO 3·2

Si è visto come il computer controlla i colori dei singoli caratteri sullo schermo. È però anche in grado di modificare le tinte della cornice esterna e dello sfondo contro cui compaiono i simboli. Non sono previsti tasti dedicati al controllo di questi colori e il modo per modificarli è quello di impartire particolari comandi.

Battere COLOR4,5. Verificare che sia scritto correttamente e, se necessario, correggere (per il comando è necessario usare la forma americana "COLOR" invece della forma inglese "COLOUR"). Battere quindi il tasto RETURN La cornice dello schermo TV diventerà immediatamente porpora.

Questo comando è costituito da tre parti:

COLOR: Questa è chiamata parola chiave.

4: Questo numero indica al computer quale parte dello schermo deve essere influenzata dal cambiamento. Così
COLOR0,...modifica lo sfondo dello schermo,
COLOR1 si riferisce al cursore, e
COLOR4 viene usato per la cornice.

5: Questo è un codice colore. La tabella completa è

Col.	Nero	Bianco	Rosso	Viola
Cod.	1	2	3	4
Col.	Porpora	Verde	Blu	Giallo
Cod.	5	6	7	8
Col.	Arancio	Marrone	Gial.-Ver.	Rosa
Cod.	9	10	11	12
Col.	Verde Gr.	Azzurro	Blu Scuro	Verde Ch.
Cod.	13	14	15	16

Si può ora comprendere perché il comando "COLOR4,5" abbia cambiato in porpora il colore della cornice. Provare altri comandi dello stesso tipo, come

COLOR4,2 RETURN

COLOR4,14 RETURN

Lo sfondo dello schermo può essere modificato allo stesso modo usando COLORØ,... Per esempio, provare

COLORØ,8

Se si commette un errore quando si batte un comando COLOR, ci si può aspettare uno dei seguenti messaggi

?SYNTAX ERROR

se il comando non è scritto in maniera corretta o

?ILLEGAL QUANTITY ERROR

se i numeri utilizzati per il comando non sono compresi nella gamma corretta (Ø-4 per il primo numero e 1-16 per il secondo).

Se si esegue un disegno, è consigliabile iniziare impostando i colori della cornice e dello sfondo tramite l'uso di appropriati comandi COLOR e quindi cancellando lo schermo con CLR HOME e SHIFT

Se si imposta lo stesso colore sia per lo sfondo sia per il cursore, il cursore sarà invisibile. Certe altre combinazioni (come, ad esempio, giallo e arancio) rendono impossibile distinguere i caratteri dallo sfondo. Se il cursore scompare alla vista dell'utente, si può sempre recuperare premendo il tasto RESET.

Esercizio 3.2 completato

ESERCIZIO 3·3

È probabile che si siano notate alcune incomprensibili mancanze fra i caratteri grafici; per esempio si trova ▚ ma non ▞ ; si trovano ◸ e ◹ ma non ◢ o ◣. Per di più, sembra che non esista la possibilità di riempire di colore un quadrato completo e quindi creare vaste aree della stessa tinta.

A questo proposito può essere d'aiuto la funzione di "inversione campo".

Quando un carattere è visualizzato in inversione di campo, i colori dei caratteri e dello sfondo vengono invertiti. Provare l'esempio seguente:

Reinizializzare il computer e battere alcuni caratteri, tra cui ▚ , ◸ , ◹ e uno spazio.

Tenere ora premuto CTRL e il tasto 9 (contrassegnato anche RVS ON).

Quindi rilasciare CTRL e battere alcuni altri caratteri. Essi appariranno in bianco su sfondo nero, invece che nero su bianco. In particolare ▚ diventa ▞ ; ◸ e ◹ vengono modificati in ◢ e ◣ e uno spazio appare come un blocchetto di colore nero. Si dice che il computer si trova nella modalità di "campo invertito".

Per riportare i caratteri nella modalità normale, tenere premuto CTRL e battere il tasto Ø (contrassegnato RVS OFF).

Il cursore non dà indicazioni riguardo alla modalità in cui il computer si trova in quel momento (normale o invertita). Se si stanno utilizzando molti simboli invertiti è facile dimenticare come apparirà il carattere successivo.

Questa difficoltà può essere sormontata in due modi diversi:

a) Battere il carattere successivo e controllarlo. Se è sbagliato, cancellarlo, cambiare modalità e provare di nuovo.

b) Battere il tasto RVS ON o, a seconda dei casi, RVS OFF, prima di battere il simbolo successivo. Se il computer si trova già nella modalità corretta ciò non avrà alcun effetto.

Un metodo efficiente per riempire lo schermo con blocchetti colorati è quello di usare spazi invertiti. La barra spaziatrice è un tasto ad effetto ripetuto e, se si tiene premuta, genererà una sequenza di circa 1Ø spazi al secondo.

Un ottimo modo per esercitarsi nell'uso del colore può essere, ad esempio quello di eseguire i disegni delle bandiere di alcune nazioni. Il tipo di bandiera più semplice da riprodurre è quello con strisce orizzontali, come quelle dela bandiera del Lussemburgo.

porpora
giallo
verde

Per eseguire il disegno di questa bandiera, impostare la cornice su un colore appropriato (ad esempio il nero) ed il colore dello sfondo sullo stesso colore dell'angolo in basso a destra della bandiera (in questo caso verde). Per far ciò, battere

COLOR4,1 RETURN

COLORØ,6 RETURN

Quindi spostare il cursore nella posizione "home" e cancellare il contenuto dello schermo, selezionare "porpora" e modalità invertita (tenere premuto CTRL e battere 5 PUR, quindi RVS ON). Quindi tenere premuta la barra spaziatrice e riempire 8 righe con spazi invertiti color porpora.

Selezionare "giallo" e riempire 9 righe con spazi invertiti color giallo.

Infine, impostare il colore su verde. Questa operazione farà scomparire il cursore e lascerà un'area verde di 8 righe nella parte inferiore della bandiera.

È importante impostare il colore dello sfondo su uno dei colori che appaiono nel disegno che si sta eseguendo, altrimenti non sarà possibile nascondere il cursore quando il disegno viene completato. Allo stesso modo il colore della cornice dovrebbe essere diverso da qualsiasi colore che appare nella bandiera.

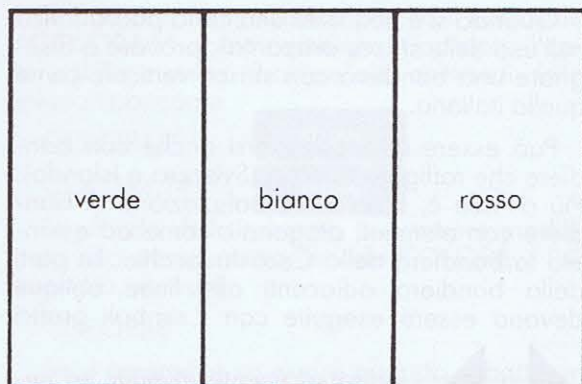
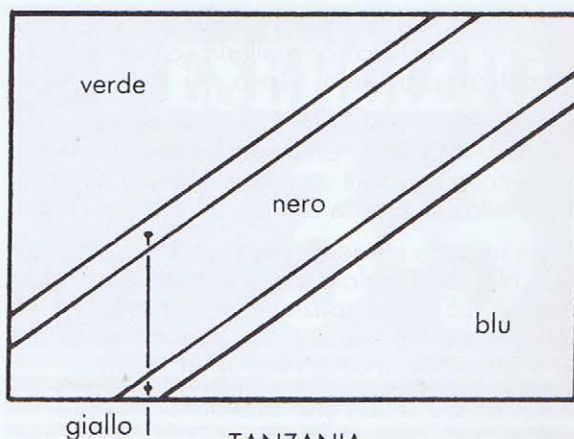
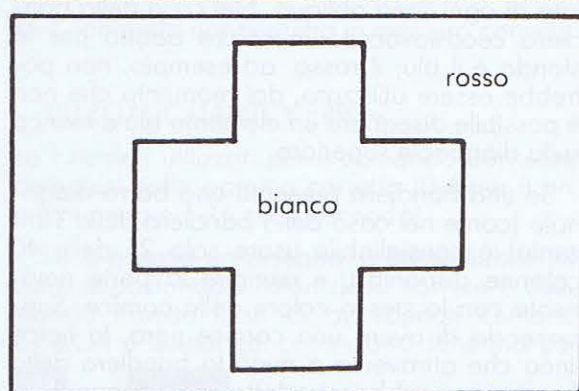
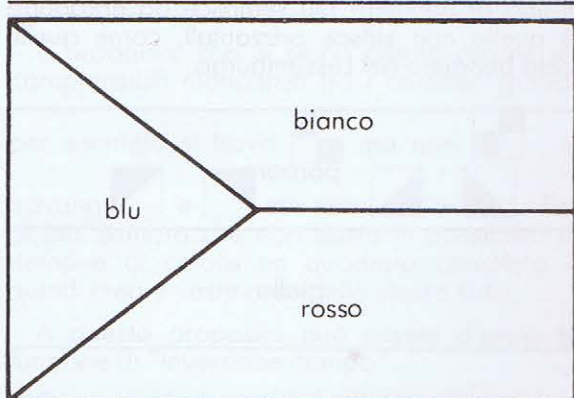
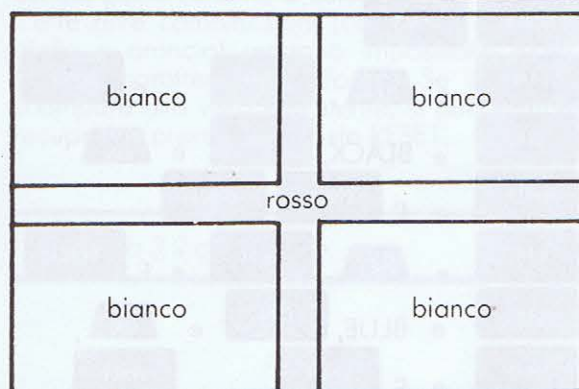
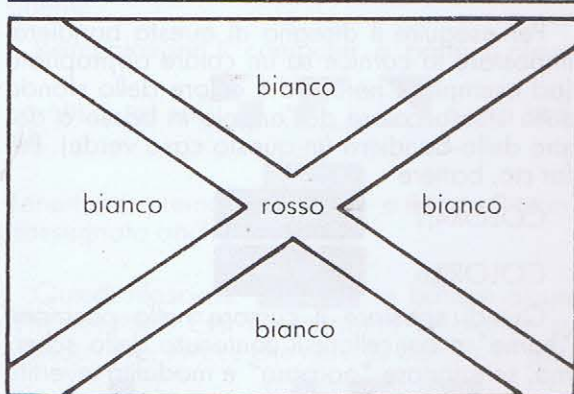
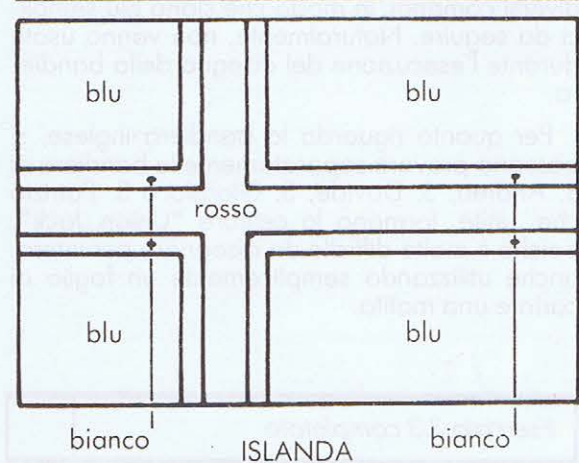
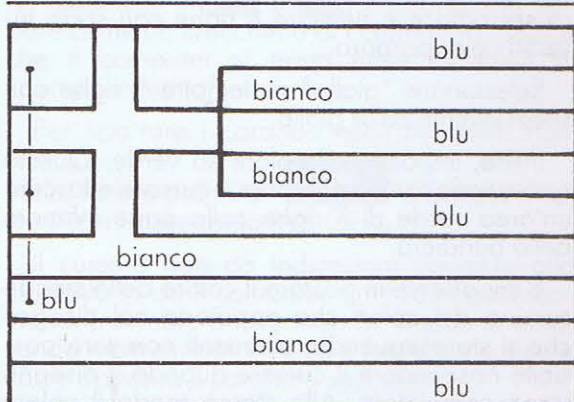
Quando si è acquisita una certa padronanza nell'uso delle strisce orizzontali, provare a disegnare una bandiera con strisce verticali, come quella italiana.

Può essere utile esercitarsi anche con bandiere che raffigurano croci (Svizzera o Islanda). Più difficile è, invece, la realizzazione di bandiere con elementi diagonali, come ad esempio la bandiera della Cecoslovacchia. Le parti della bandiera adiacenti alle linee oblique devono essere eseguite con i simboli grafici o , adeguatamente invertiti, se necessario. Se si considera attentamente la questione, si vedrà che il colore dello sfondo deve essere scelto in modo tale che si trovi da un lato di ogni linea obliqua. Nel caso della bandiera cecoslovacca, un colore adatto per lo sfondo è il blu; il rosso, ad esempio, non potrebbe essere utilizzato, dal momento che non è possibile disegnare un elemento blu e bianco sulla diagonale superiore.

Se una bandiera presenta una barra diagonale (come nel caso della bandiera della Tanzania) è consigliabile usare solo 25 delle 40 colonne disponibili, e riempire la parte rimanente con lo stesso colore della cornice. Supponendo di avere una cornice nera, la tipica linea che attraversa a metà la bandiera della Tanzania verrebbe introdotta come segue:

Le virgole, così come la parola "e", sono state utilizzate esclusivamente per separare i diversi comandi, in modo che siano più semplici da seguire. Naturalmente, non vanno usate durante l'esecuzione del disegno della bandiera.

Per quanto riguarda la bandiera inglese, si possono provare separatamente le bandiere di S. Andrea, S. Davide, S. Giorgio o S. Patrizio che, unite, formano la celebre "Union Jack", poiché è molto difficile da disegnare per intero, anche utilizzando semplicemente un foglio di carta e una matita.

Esercizio 3.3 completato	

ESERCIZIO 3·4

Un'altra utile funzione del COMMODORE **16** e del **PLUS/4** è quella dei caratteri lampeggianti. Qualsiasi carattere può essere fatto lampeggiare, come il cursore. Il lampeggiamento viene controllato da due tasti, contrassegnati "Flash on" e "Flash off", che si trovano vicino alla barra spaziatrice.

Questi tasti vengono usati quasi nello stesso modo di RVS ON e RVS OFF . Per avere un esempio, provare a battere la sequenza

VOTATE e FLASH GORDON e CTRL PER LA PRESIDENZA

Eseguire un disegno colorato, usando l'intera gamma dei colori forniti dal computer. Questa operazione può essere facilitata disegnando la figura che si intende visualizzare prima su un foglio di carta a quadretti, usando delle matite colorate.

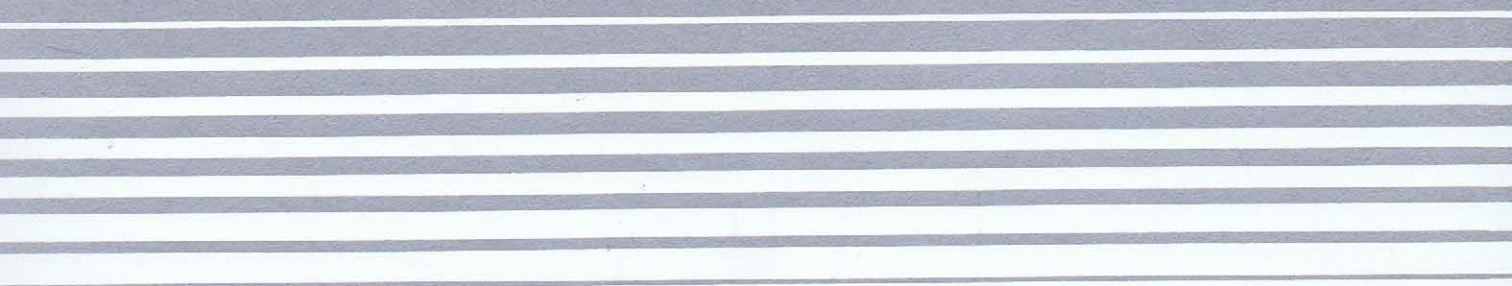
Esercizio 3.4 completato	

Il programma che corrisponde alla Lezione 3 è un quiz, e può essere caricato battendo:

```
LOAD"UNIT3QUIZ"
```

oppure

```
OPEN1,8,15,"I"
LOAD"UNIT3QUIZ",8
```

LEZIONE: 4

Nelle prime tre Lezioni del corso è stata presa in esame quasi interamente la tastiera del computer e il suo uso per visualizzare un testo e disegnare immagini sullo schermo del televisore. Questa è una buona preparazione per la successiva parte del corso dove verranno trattate alcune delle funzioni che il computer può eseguire su comando dell'operatore.

Come già noto, il computer eseguirà varie azioni quando gli verrà ordinato di farlo. I necessari comandi sono scritti in BASIC, un semplice ed diffuso linguaggio per computer ideato da Kemeny e Kurtz del Dartmouth College, USA. Il BASIC ha le proprie regole di grammatica esattamente come un qualsiasi altro linguaggio, ma fortunatamente esse sono semplici da imparare e verranno facilmente ricordate con la pratica, senza nessuno sforzo particolare.

Qualsiasi comando BASIC inizia con una "parola chiave" tipo LOAD o POKE o PRINT. Ciò comunica al computer il tipo di comando.

Analogamente ogni comando termina con il tasto RETURN. Questo ha due significati diversi:

se la parola chiave è la prima parola sulla riga, RETURN si può paragonare al colpo di pistola dello starter: "Pronti! Via". Per esempio se si batte

LOAD"TESTCARD"

il caricamento effettivo inizia quando viene premuto il tasto RETURN. L'altra interpretazione di RETURN è trattata nella Lezione seguente, ma eccone un breve anticipo:
se la parola chiave in un comando BASIC è preceduta da un numero, ad esempio

3 PRINT 5+7

il tasto RETURN segnala di non ubbidire al comando ma di memorizzarlo per uso successivo. In questa Lezione verrà presa in esame la prima interpretazione. I nostri comandi non saranno preceduti da numeri e RETURN sarà un segnale al computer di intraprendere un'azione immediata.

ESERCIZIO 4·1

Uno dei comandi più utili e flessibili è PRINT. Esso fa sì che il computer esegua qualcosa e visualizzi il risultato sullo schermo. La parola PRINT è usata in quanto il sistema originale BASIC installato a Dartmouth si basava su telescriventi che effettivamente "stampavano" le risposte su rotoli di carta.

L'Esercizio 4.1 è organizzato in 3 fasi. Nella prima si proveranno alcuni comandi PRINT, prendendo nota dei risultati. Nella seconda saranno indicate le caratteristiche indicate negli esempi. Nella terza verranno esaminati alcuni nuovi comandi PRINT, cercando di prevedere il loro effetto sul computer. Le risposte possono essere controllate tramite computer stesso.

Iniziare battendo questi comandi e terminando ciascuno di essi con il tasto RETURN. Assicurarsi di impostare i comandi correttamente usando i tasti di controllo del cursore per correggere eventuali errori di battitura. Prendere nota accurata delle risposte nelle apposite caselle. Le prime due caselle sono già state compilate:

PRINT 999	999 READY.
PRINT"CIAO"	CIAO READY.
PRINT –56	
PRINT 3+4+4	
PRINT 5★7	

PRINT 27/7	
PRINT "COMPUTER"	
PRINT VIC	
PRINT 3,5	
PRINT 3;5	
PRINT"CONIGLIO","CANE"	
PRINT"PESCE";"GATTO"	
PRINT"3+5"	
PRINT29–12;"LEONI"	
PRINT 1;2;3;4	

Prima di procedere, studiare le note accuratamente per rendersi conto di quante diverse caratteristiche di PRINT è possibile ottenere. Un aspetto comune delle risposte è che tutte sono seguite da READY. Ma ciò è vero per qualsiasi comando il cui risultato appare direttamente sullo schermo, non si tratta quindi di una caratteristica specifica di PRINT. Ecco i punti principali riguardanti il comando PRINT:

1. Il comando può gestire sia numeri sia stringhe e lo fa in modi diversi: un numero può essere indicato esplicitamente (ad esempio 999) oppure sotto forma di un'espressione o "somma" che il computer elabora. Dalle espressioni riportate negli esempi 3+4+4, 5★7, 27/7 e 29–12, si può vedere che il computer ha la possibilità di sommare, sottrarre, moltiplicare e dividere. I segni ★ e / significano rispettivamente moltiplicazione e divisione. (Se si è interessati ad usare il computer per calcoli matematici avanzati, è possibile utilizzare espressioni più complete che prevedono l'uso della parentesi e di tutte le funzioni speciali utilizzate nel calcolo scientifico. Si consiglia di osservare l'Appendice A che è un'unità extra intesa particolarmente per questo scopo).

   Una *stringa* è qualsiasi sequenza di caratteri racchiusa tra virgolette. Il comando PRINT ripete tale stringa esattamente come è stata indicata, senza cercare di elaborarla in alcun modo. Le stringhe del nostro testo erano

   "CIAO", "COMPUTER", "CONIGLIO" "CANE", "PESCE", "GATTO", "3+5" e "LEONI".

   Notare che "3+5" non è un'espressione quantunque possa sembrare tale; è chiusa tra virgolette e quindi viene trattata come una stringa. Se si vuole visualizzare una stringa ma si dimentica di racchiuderla tra virgolette, probabilmente (ma non sempre) si otterrà 0.

2. Il comando PRINT può gestire due o più stringhe contemporaneamente. Se due stringhe sono separate da virgole, il secondo risultato è ben distanziato sullo schermo (esattamente sull'altra metà). Se viene usato un punto e virgola, la separazione è minore. In casi particolari la separazione è nulla (come nell'esempio "PESCEGATTO"). I numeri visualizzati dal computer appaiono separati in quanto ogni numero è sempre preceduto da uno spazio (oppure da un segno meno se è negativo) e sempre seguito da un altro spazio. Se ciò non risulta chiaro, ripetere il comando

   PRINT 1;2;3;4;

   e 'misurare' il risultato tramite il cursore.

3. Verranno ora presi in esame gli spazi all'interno del comando stesso. La parola chiave PRINT deve essere compatta (cioè le lettere non devono essere separate da spazi) e qualsiasi spazio all'*interno* di una stringa è parte integrante della stringa stessa e verrà riprodotto. Diversamente, gli spazi tra stringhe o tra numeri verranno ignorati.
   Pertanto

   PRINT 3 +5;7; 8

   darebbe lo stesso risultato che

   PRINT3+5;7;8

   Questa regola – che gli spazi nei comandi sono ignorati ovunque salvo nei comandi delle parole chiave e nelle stringhe – è generalmente valida per tutto il BASIC.

4. Se si compie un errore, nella parola chiave, oppure se si scrive un'espressione che non ha senso (ad esempio 5★★7), il computer respinge il comando con il commento

?SINTAX ERROR

Questo è il gergo del computer per dire che sono state infrante le regole del BASIC. Non c'è nulla da fare salvo correggere il comando e provarlo di nuovo.
Procedere ora con la seguente lista di comandi PRINT cercando di prevedere l'effetto che ciascuno di essi avrà sul computer. Indicare le spaziature previste all'interno dei risultati: esse sono importanti quanto i risultati stessi. Non è invece necessario scrivere ogni volta READY..
Attenzione: alcuni comandi contengono deliberatamente degli errori.
Controllare le risposte sul computer. Se sono stati fatti degli errori di cui non si riesce ad identificare la causa, ripetere l'intero esercizio fino alla eliminazione del problema.

Ad esempio:

PRINT 5+2★7 darà 19

e

PRINT 5+2+6/3–3 darà 6

Comando	*Previsione*	*Risultato del computer*
PRINT 94		
PRINT 8–5		
PRINT 3★2+5		
PRINT "PERBACCO"		
PRINT ABBASTANZA		
PRINT "1★/3"		
PRINT 3;47		
PRINT 2+2;2–2;2★2;2/2		
PRINT "RAFFICA DI"; "PIOGGIA"		
PRINT 8–7		
PRINT "18","TOPI"		
PRINT 53+★7		
PRINT –1;–2;–3		

Esercizio 4.1 completato	

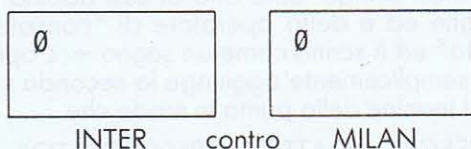
ESERCIZIO 4·2

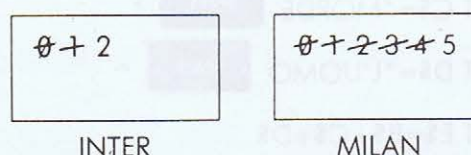
Se un computer potesse eseguire un solo comando alla volta non sarebbe di grande utilità. Avrebbe cioè la stessa funzione di una normale calcolatrice (non programmabile).

La caratteristica più utile del funzionamento di un computer consiste nella possibilità di eseguire intere sequenze di comandi, denominate "programmi". Mano a mano che i comandi della sequenza vengono eseguiti, devono essere previsti dei meccanismi per la registrazione dei risultati intermedi, per la memorizzazione del punto a cui il programma è giunto e per la trasmissione dei risultati da un comando a quello successivo. La memoria che serve per collegare i comandi è fornita sotto forma di *variabili*.

Prima di trattare espressamente le variabili BASIC, si cercherà di introdurre il concetto con un esempio riguardante una partita di calcio in cui il segnapunti deve eseguite le seguenti operazioni:

Prima che la partita inizi, disegnare due riquadri, contrassegnarli con il nome delle squadre e scrivere degli zero all'interno, così:

INTER contro MILAN

Ogniqualvolta una delle due squadre segna un goal, aggiornare il punteggio nella casella corrispondente, aumentandolo di una unità. Cancellare o barrare il vecchio numero. Ad un certo punto, durante la partita la situazione potrebbe essere

INTER MILAN

Al fischio finale, usare la lavagna per visualizzare i nomi delle squadre unitamente ai numeri (più recenti) all'interno dei riquadri e cioè:

INTER	2
MILAN	5

In questo esempio le caselle o riquadri sono *le variabili*. I numeri nelle caselle servono per indicare il risultato della partita "minuto per minuto"; ma le etichette (INTER e MILAN) rimangono le stesse per la durata dell'incontro. Questo meccanismo è semplice ma efficace. La memoria del computer (costituita come si è detto da 60671 o 12277 byte) si può in un certo qual modo paragonare ad una lavagna. Al momento dell'accensione del computer, la lavagna viene completamente cancellata. In seguito, quando una variabile viene nominata per la prima volta, il computer "disegna una casella", riservando una porzione di memoria e la contrassegna con il nome scelto dall'utente. Quindi "scrive un numero nella casella" memorizzando il valore appropriato nella porzione di memoria che è stata riservata. Il comando BASIC che realizza questa operazione, contiene la parola chiave LET. Esaminiamo questo comando in dettaglio:

LET X=5

Qui il nome della variabile è X. Il computer creerà una casella denominata X (se non lo ha già fatto) e vi inserirà il numero 5. Se esiste già una variabile X, non viene riservato altro spazio; il 5 semplicemente *sostituisce* il valore precedente. Osservare i casi seguenti:

	X non esiste *(Caso 1)*	X esiste *(Caso 2)*
Prima	(Memoria vuota)	[37] X
Dopo	[5] X	[5] X

Risultato di LET X = 5

Quando si impartisce un comando LET, il computer risponde semplicemente

READY.

Non c'è traccia sullo schermo di ciò che la macchina ha fatto. Fortunatamente, si può utilizzare il comando PRINT per visualizzare il valore di una variabie ogniqualvolta è citata per nome. Provare la seguente sequenza di comandi:

	Risultato
LET Z = 14 PRINT Z	
LET Z = 31 PRINT Z	

Se si mantiene l'ordine esatto, il primo valore di Z da stampare sarà 14 e il secondo 31. Il primo comando LET crea una variabile denominata Z e le attribuisce il valore 14; il secondo semplicemente ne cambia il valore in 31.

A questo punto è opportuno considerare alcune semplici regole sulle variabili e sui loro nomi.

In BASIC ci sono due tipi di variabili:

- Variabili numeriche, per memorizzare i numeri.
- Variabili stringa per memorizzare stringhe (ad esempio parole o frasi)

La scelta dei nomi per le variabili è abbastanza limitata. Una variabile numerica può essere richiamata da una singola lettera, da una lettera seguita da una cifra o da due lettere. Alcuni esempi di nomi possibili per le variabili numeriche sono:

A,X,Z,B5,TX,PQ

I nomi per le variabili stringa terminano sempre col segno $. Salvo ciò le regole per le variabili stringa sono le stesse di quelle per le variabili numeriche. Gli esempi sono:

C$,Z$,P7$,DB$

Per mostrare l'uso delle variabili stringa, provare a battere

```
LET T$="BUON"
PRINT T$;"GIORNO"
```

Il valore che segue il segno = in un comando LET non deve necessariamente essere un semplice numero o una stringa; può essere un'*espressione* e inoltre esso può usare i valori correnti delle variabili facendo riferimento ai loro nomi. Per esempio, osservare la sequenza di comandi:

```
LET Q=5
LET S=Q+3
```

Il primo crea una variabile denominata Q (è una variabile numerica come si può vedere dal nome) e imposta il suo valore a 5. Il secondo crea una variabile denominata S, le assegna il valore Q, aggiunge 3 e pone il risultato in S. Per verificare questo punto, provare ad eseguire questi due comandi ed esaminare il risultato battendo

```
PRINT Q;S
```

Si osservi ora la seguente sequenza e si *preveda* il risultato dell'istruzione PRINT in ciascun caso:

LET AA = 15 LET B = 33—AA PRINT AA,B	
LET D = 3 LET E = D★D+7 LET F = E—D PRINT F;E;D	
LET F=4 LET F=F+1 PRINT F	

L'ultima era esatta? Alcuni potrebbero trovarla un po' complicata.

Non c'è limite al numero di diversi valori che una variabile può assumere purchè ne contenga uno alla volta. Un comando tipo

LET F=F+1

significa: Per *prima* cosa calcolare l'espressione (prendendo il valore di F e aggiungendo 1)
Quindi inserire il risultato nella casella F, sostituendo il valore precedente.

In altre parole, il comando fa sì che il computer aggiunga 1 al valore corrente di F.

I segni che consentono di combinare i numeri in vari modi sono detti *operatori aritmetici*. Essi sono +,–,★ e /. Il BASIC consente inoltre di manipolare stringhe in vari modi usando gli operatori stringa. Solo uno di essi abbina due stringhe ed è detto operatore di "concatenamento" ed è scritto come un segno +. L'operatore semplicemente aggiunge la seconda stringa al termine della prima in modo che

"PESCE"+"GATTO"="PESCEGATTO"

Osservare la seguente sequenza di comandi e prevedere il risultato delle istruzioni PRINT. Quindi provare la sequenza sul computer; ricordarsi di lasciare uno spazio prima di ciascuna virgoletta di chiusura:

LET E$=B$+C$+D$

LET F$ =D$+C$+B$

PRINT E$ [ ]

PRINT F$ [ ]

PRINT e LET sono i due comandi più frequentemente usati nel BASIC. Vale la pena di ricordare che quando si usa il computer è possibile sostituire la parola PRINT con un simbolo: il punto di domanda (?). LET può essere omesso del tutto. Una sequenza valida è

```
A=5
B=17
?A,B
```

Il programma che viene fornito con questa Lezione è studiato per fare abbondante pratica con i comandi PRINT e LET ed è detto UNIT4DRILL.
È possibile interrompere il programma quando si è sicuri di aver perfettamente compreso l'uso delle variabili numeriche e stringa.

Esercizio 4.2 completato	

LEZIONE: 5

È venuto il momento di esaminare i comandi memorizzati. Si inizierà dimostrando che il computer può effettivamente conservare i comandi in memoria, richiamandoli in un secondo tempo. Accendere il computer (oppure se sta già eseguendo un altro programma, ad esempio SPEEDTYPE, arrestarlo battendo RUN STOP) e impartire il comando NEW (seguito come al solito dal tasto RETURN).

Questo comando cancella la memoria del computer esattamente come un insegnante cancella la lavagna all'inizio della lezione. Non succederà nulla, salvo la comparsa della risposta READY, in quanto la memoria è tutta all'interno del computer.

Ora battere il comando contrassegnato

10 PRINT 13+59

(NOTA: battere "uno zero" e non la O maiuscola)

e premere RETURN . Il solo risultato visibile è che la macchina sposta il cursore all'inizio della riga successiva. Il risultato della somma 13+59* non è calcolato nè visualizzato sullo schermo. Per contro, è successo qualcosa di invisibile: il computer ha ricordato il comando e lo ha immagazzinato nella sua memoria interna.

Per verificare ciò, cancellare per prima cosa lo schermo (usando i tasti SHIFT e CLR HOME) e quindi impartire il comando

LIST

Se l'operazione è stata eseguita correttamente, sullo schermo dovrebbe ricomparire una copia del comando interessato. Ciò dimostra che è stato conservato nella memoria del computer. Un altro modo per impartire il comando LIST consiste nel tenere premuto il tasto SHIFT contemporaneamente al tasto lungo contrassegnato HELP f 7 .

Finora, il comando PRINT è stato memorizzato e richiamato ma non è stato ancora eseguito.

Il computer deve ancora effettuare la somma 13+59. Per calcolare il risultato, battere

GOTO 10

ricordando di usare le lettere O maiuscole (e non le cifre 0) nel comando GOTO. Ciò comunica al computer di eseguire il comando contrassegnato "10". A questo punto compare la risposta. È possibile eseguire questa operazione quante volte si vuole. Un comando non viene distrutto quando viene listato o eseguito.

Il computer ricorda molti comandi contemporaneamente. (Il limite è definito dalla dimensione della memoria: occorre un byte per contenere ciascun carattere di un comando più qualche bit di supervisione per il comando nel suo complesso). Ogni comando deve essere preceduto dalla propria etichetta o contrassegno e tutte le etichette devono essere diverse. La macchina memorizza e lista sempre i comandi in ordine crescente di etichette ed esegue questi comandi nell'ordine, a meno che non venga specificato un ordine diverso.

Provare battendo NEW

```
10 PRINT"PRIMA RIGA"
20 A=5
30 B=10
40 PRINT A;B;A+B
50 STOP
```

Ricordarsi di terminare ciascun comando con RETURN

Provare ora un LIST e quindi un GOTO 10 e controllare che i risultati siano quelli previsti. Il comando STOP interrompe il computer e visualizza un READY quando raggiunge la fine della sequenza dei comandi.

La sequenza in cui i comandi sono memorizzati è quella esatta anche se li si batte in ordine diverso dai rispettivi numeri di etichetta. Per esempio, se si fosse battuto

```
30 B=10
10 PRINT"PRIMA RIGA"
40 PRINT A;B;A+B
50 STOP
20 A=5
```

questi cinque comandi sarebbero sempre stati listati ed eseguiti nell'ordine 10, 20, 30, 40, 50. Cancellare la memoria con un NEW e provare ancora.

* *Non c'è nulla di speciale a proposito della somma 13+59. Qualsiasi altro comando PRINT avrebbe funzionato altrettanto bene in questo esempio.*

Iniziare sempre con i numeri che vanno a incrementi di 1Ø. Se si decide successivamente di inserire qualche comando tra quelli già scritti, l'operazione sarà possibile usando numeri di etichette intermedie come 15 o 38.

Perché preoccuparsi di memorizzare i comandi? Ci sono due buoni motivi:

- I comandi che vengono richiamati dalla memoria interna del computer sono eseguiti più velocemente di quelli che vengono battuti.
- I comandi che sono stati battuti una volta, possono essere eseguiti parecchie volte. Praticamente, ogni lavoro utile eseguito dal computer, comporta la ripetizione e vale quindi la pena di inserire i comandi nella memoria del computer da cui vengono richiamati facilmente e velocemente.

Forse il modo più facile per ottenere la ripetizione è di memorizzare un comando contrassegnato GOTO. Si consideri il seguente programma:

```
1Ø PRINT"NORD"
2Ø PRINT"OVEST"
3Ø PRINT"SUD"
4Ø PRINT"EST"
5Ø GOTO 1Ø
```

Quando viene avviato dall'etichetta 1Ø, il computer obbedisce ai primi quattro comandi in sequenza. Il successivo comando lo rimanda all'etichetta 1Ø, cosicchè inizia la sequenza da capo. Esso continua così all'infinito e si ferma soltanto battendo RUN STOP o spegnendo il computer.

Cancellare ora la memoria e battere il programma. Avviarla dando il comando iniziale

GOTO 1Ø

Si vedrà la macchina obbedire alle righe del programma, molto più velocemente di quanto non sia possibile leggerle. È possibile rallentare la macchina premendo e tenendo abbassato C= oppure è possibile fermarla nel modo solito con il tasto RUN STOP.

A questo punto si vedrà effettivamente il vantaggio di usare i numeri di etichetta distanziati di 1Ø. Supponiamo ora di voler variare il programma in modo che comprenda le direzioni diagonali

NORD
NORD-OVEST
OVEST
SUD-OVEST
ecc.

Occorrono in questo caso nuove istruzioni da inserire tra quelle esistenti. Se le si numera 15, 25, 35 e 45 esse verranno inserite nelle posizioni giuste. Battere quanto segue:

```
15 PRINT"NORD-OVEST"
25 PRINT"SUD-OVEST"
35 PRINT"SUD-EST"
45 PRINT"NORD-EST"
```

Ora listare (con un comando LIST) il programma e controllare che le nuove righe siano state inserite tra le vecchie nelle posizioni giuste. Eseguire il programma e osservare il risultato.

Ora scrivere e provare un programma con gli stessi numeri di etichetta. Se nell'ambito delle stringhe si utilizzano caratteri grafici anzichè alfabetici, si possono ottenere sullo schermo risultati interessanti.

Il comando GOTO 1Ø che è usato per avviare il programma, ha un equivalente molto più comodo il comando, RUN. Questo fa sì che il computer inizi ad eseguire i comandi partendo da quello con numero minore. Un ";" dopo una stringa impedisce al computer, durante l'esecuzione, di andare a capo dopo la visualizzazione della stringa stessa. (Naturalmente occorre sempre terminare ciascun comando con il tasto RETURN). Per contro, esso inizia una nuova riga sullo schermo soltanto quando raggiunge il margine di destra. Un semplice programma come quello che segue riempirà rapidamente l'intero schermo con disegni particolari; provarlo e cercare di comprendere il meccanismo.

```
1Ø PRINT "├┤┬┴ ┌┐ └";
2Ø GOTO 1Ø
```

Esercizio 5.1 completato	

Una sequenza di comandi che viene ripetuta più volte, è detta iterazione o loop. Un'iterazione può includere diversi tipi di comandi compreso un LET che attribuisce un nuovo valore ad una variabile. Osservare questo programma e cercare di prevederne il funzionamento:

```
10 LET A=1
20 PRINT A
30 LET A=A+1
40 GOTO 20
```

Simulare il comportamento del computer eseguendo passo per passo le operazioni indicate dal programma. Prendere nota delle nozioni del valore della variabile A. Non procedere con la lettura fino a che non si è riflettuta e scritta la risposta.

(e così via)

Immettere ora il programma ed eseguirlo, tenendo abbassato il tasto C= per rallentarlo.

(Ma non toccare C= fino a che non si è battuto RETURN dopo RUN).

Nonostante la semplicità del problema, se ne dà qui di seguito una breve spiegazione.

Il programma inizia eseguendo il comando contrassegnato 10, che dà alla variabile A il valore 1. Il successivo comando visualizza questo valore sullo schermo.

Il comando 30 sostituisce A con A+1. Questo equivale ad aggiungere 1 al vecchio valore di A, cosicchè il risultato (questa volta) è 2. Il successivo comando è un GOTO e fa sì che il computer torni al comando 20. La macchina continua a lavorare sull'intera sequenza, 20, 30, 40, ripetutamente, ma ogni volta il valore di A viene aumentato di 1. Ciò produce la sequenza 1,2,3...

Per fare un po' di pratica, cercare di prevedere le prime righe visualizzate da questi due programmi (ricordarsi che ★ significa moltiplicato per):

10 B=0	**10 A = 1**
20 PRINT B	**20 B = A★A**
30 B = B+3	**30 PRINT A,B**
40 GOTO 20	**40 A=A+1**
	50 GOTO 20

E ora controllare la correttezza del risultato.

È possibile applicare lo stesso meccanismo anche alle stringhe. Provare questo programma:

```
10 X$="★"
20 PRINT X$
30 X$=X$+"#"
40 GOTO 20
```

I successivi valori di X$ mano a mano che il programma procede nell'iterazione saranno ★,★#,★##,★###, e così via. La stringa X$ diventa sempre più lunga e usa sempre più spazio sullo schermo ogni volta che viene stampata.

Dopo circa 45 secondi, la lunghezza della stringa supera la capacità di memorizzazione del computer (il numero massimo di caratteri ammesso è 255) e viene quindi segnalata una situazione anomala:

```
?STRING TOO LONG
ERROR IN 30
```

Il messaggio ERROR in 30 significa che il comando che ha tentato di memorizzare la stringa errata era quello contrassegnato 30.

Ecco alcuni altri programmi sui quali fare previsioni.

10 A$ = "++"	**10 A$ = "XY"**
20 PRINT A$	**20 PRINT A$**
30 A$ ="A"+A$+"—"	**30 A$ = A$+A$**
40 GOTO 20	**40 GOTO 20**

Ricordare che se una lettera compare all'interno di una stringa, si tratta semplicemente di una lettera e non del nome di una variabile. Così "X" non ha nulla a che fare con la variabile X o X$.

Come esercizio finale, scrivere un programma contenente una semplice iterazione ed eseguirlo esattamente per 1 minuto, controllando col cronometro. Quindi fermarlo e calcolare in base al numero delle iterazioni il numero totale dei comandi eseguiti. Da questo, si risalirà facilmente al numero di comandi eseguiti in 1 secondo.

Il quiz di auto-test per questa lezione è detto UNIT5QUIZ

Esercizio 5.2 completato	

LEZIONE: 6

L'obiettivo di questo intero corso è quello di facilitare l'apprendimento della progettazione di programmi propri. A questo scopo, viene presentata una raccolta di tecniche o "strumenti" che permettono una migliore organizzazione del lavoro, e la possibilità di correggere eventuali errori.

Questa Lezione può essere considerata come una "scatola di attrezzi" o una "cassetta di pronto soccorso". Non vi parlerà questa volta di programmazione come tale ma i suoi contenuti saranno utili in caso di emergenza. Leggere la Lezione attentamente, cercare di apprendere le tecniche che essa descrive e tenerle sempre presenti mano a mano che si procede nel corso.

Se si dispone di un'unità a dischetti, è necessario inizializzare un nuovo dischetto prima di iniziare l'Esercizio 6.1.

Ovviamente, quanto segue non riguarda gli utenti che utilizzano un registratore a cassetta per caricare il programma. Com'è noto, su un disco si possono memorizzare diversi programmi, inoltre sul disco è presente anche un indice o 'catalogo' che elenca i nomi e le dimensioni dei programmi memorizzati su quel disco.

Provare col seguente Esercizio. Caricare il disco programma fornito insieme al corso e battere:

DIRECTORY RETURN

Questo comando richiama nel computer il catalogo o indice del disco e lo visualizza sullo schermo, che si presenta più o meno come segue:

```
0 "ITB PROGRAMS         " DT 2A
12 "TEST PROG"            PRG
10 "HANGMAN"              PRG
13 "SPEEDTYPE"            PRG
```

e così via fino a

470 BLOCKS FREE

Vale la pena di studiare questo catalogo. La riga superiore, che compare in negativo, fornisce l'identità del disco stesso. In questo caso il nome –ITB PROGRAMS– è stato scelto dalla Commodore. "DT2A" è il numero di matricola del disco.

Ognuna delle righe successive riguarda un programma. Il primo elemento fornisce la dimensione del programma in blocchi, ciascuno dei quali contiene 256 byte, cosicchè è possibile notare ad esempio, che HANGMAN ha una lunghezza in byte pari a 23x256=5888.

Il secondo elemento fornisce il nome del programma il terzo, "PRG", è identico per ogni programma.

La riga al termine del catalogo indica quanti sono i blocchi inutilizzati. Il disco ITB PROGRAMS può contenere ancora 20 programmi delle dimensioni di HANGMAN, oppure un numero superiore di programmi più piccoli. La capacità di un disco completamente vuoto è di 664 blocchi.

Un ulteriore e più veloce modo di ottenere il catalogo del disco consiste nel battere il tasto f3 f6 da solo. Ora che è stata chiarita la struttura dei dischi programma, è possibile passare alla fase successiva.

Munirsi di un disco nuovo e vuoto del tipo adatto preferibilmente acquistato dal rivenditore Commodore.

Assicurarsi che la tacca posta sulla sinistra quando si carica il disco sia libera e non coperta da un'etichetta argentata * come avviene per il disco programma ITB PROGRAMS.

Accendere l'unità a dischetti e caricare il disco. Battere

HEADER "nome disco", DØ,IØ1 RETURN

dove "nome disco" è il titolo che si vuole attribuire al proprio disco, ad esempio PROGRAMMI DI GINO.

Il titolo può avere una lunghezza massima di 16 caratteri. DØ è il numero del drive. Ciò che segue la virgola è un numero di identificazione preceduto dalla lettera I.

In questo caso si dovrebbe battere:

HEADER"PROGRAMMI DI GINO",DØ,IØ1

Attendere circa 1 minuto fino a che la spia rossa non cessi di lampeggiare sul drive e quindi battere

DIRECTORY RETURN

Si ottiene un indice o catalogo vuoto.

Sullo schermo compare cioè la scritta

```
0 "PROGRAMMI DI GINO"       01 2A
664 BLOCKS FREE
```

È stato così formattato un disco. Ogni nuovo disco che viene acquistato deve essere formattato solo una volta ma in compenso occorre inizializzarlo ogni volta che viene caricato. Se si esegue la formattazione di un disco che è già stato usato per memorizzare programmi, si distrugge tutto ciò che vi è stato registrato. Fare quindi attenzione!

Estrarre il disco così preparato dall'unità, scrivere il titolo sull'etichetta e metterlo da parte.

---

* *L'etichetta rende impossibile al drive di effettuare registrazioni sul disco e rappresenta un modo per impedire che i programmi relativi al corso vengano accidentalmente distrutti. Ciò non è ovviamente valido al momento della registrazione di un programma su un nuovo disco.*

ESERCIZIO 6·1

Caricare ed eseguire il programma della Lezione 6 denominato SENTENCES. Osservare le frasi 'casuali' che esso visualizza. Queste sono espressioni non significative e costruite mediante un artificio in cui ciascuna parola o frase viene scelta a caso da una breve lista di parole o frasi possibili. In questo caso non è importante comprendere il funzionamento del programma (anche se in linea di principio è abbastanza semplice), ma il programma stesso verrà usato come esempio per mostrare come listare, variare e conservare programmi lunghi.

Dopo aver esaminato un numero sufficiente di frasi, interrompere il programma con il tasto RUN STOP ed impartire un comando LIST. Il programma è troppo ampio per potersi inserire interamente sullo schermo per cui, man mano che la lista viene eseguita, la maggior parte di essa scompare dalla parte superiore del video. Al termine, saranno visibili gli ultimi undici comandi. Il linguaggio BASIC comprende alcune versioni speciali del comando LIST per tenere conto di questa situazione. Qui di seguito vengono elencate le cinque possibilità previste.

- È possibile listare l'intero programma battendo LIST, oppure premendo contemporaneamente SHIFT e F7. Chiaramente, ciò presenta degli inconvenienti se il programma è troppo lungo.
- È possibile listare uno specifico comando indicando il suo numero. Ad esempio

  LIST 11ØØ

  visualizza il comando 11ØØ (e nessun altro).
- È possibile listare tutti i comandi fino a un dato numero di etichette inserendo un segno – davanti al numero. Quindi

  LIST –8Ø

  mostra tutti i comandi dall'inizio del programma fino a quello contrassegnato 8Ø.
- È possibile chiedere tutti i comandi da un dato numero fino alla fine del programma inserendo un segno – dopo il numero:

  LIST 9Ø9Ø–
- Infine è possibile listare tutti i comandi compresi tra due numeri qualsiasi citandoli entrambi:

  LIST 2ØØØ – 2Ø9Ø

Ora usare alcuni di questi tipi di comandi LIST per osservare le varie parti del programma. Si noterà rapidamente che i numeri di etichetta o di identificazione non si susseguono sempre con incremento 1Ø e ciò in quanto il programma è stato modificato molte volte dopo che è stato scritto.

All'inizio del programma e in parecchi altri punti si vedranno comandi con la parola chiave REM seguita da descrizioni in inglese. REM è un'abbreviazione per REMARK (commento). Le righe che contengono commenti non svolgono alcun ruolo nel programma, ma sono incluse soltanto per rendere più facile la lettura del programma stesso. Quando si inizia a scrivere programmi complicati occorre sempre usare abbondanti REM per spiegare ciò che si sta facendo.

Una volta comprese a fondo le varie forme del comando LIST, provare a fare qualche esercizio per controllare il risultato nei seguenti casi:

(a) Il comando a cui si fa riferimento non è presente.

(provare LIST 65Ø)

(b) I numeri di etichetta sono nell'ordine sbagliato

(provare LIST 11ØØ – 1ØØØ)

(c) Il comando LIST è incluso in un programma. Battere NEW, quindi battere questo programma ed eseguirlo.

```
1Ø PRINT"PROVA LIST"
2Ø LIST
3Ø GOTO 1Ø
```

Tenere premuto il tasto C= in modo da poter vedere esattamente cosa sta succedendo. È possibile interrompere il programma con il tasto RUN STOP

Esercizio 6.1 completato	

ESERCIZIO 6·2

Questa Lezione spiega come i programmi possono essere modificati. Per cominciare, si eseguiranno modifiche ad un programma originariamente scritto da qualcun altro, ma successivamente la maggior parte delle modifiche riguarderanno programmi propri.

Ci sono tre tipi di modifiche che è possibile apportare ad un programma:

(a) Eliminazione di comandi esistenti

(b) Aggiunta di nuovi comandi

(c) Correzione o sostituzione di comandi esistenti

Eliminazione di comandi esistenti

Sono previsti otto modi per eliminare dalla memoria del computer un comando contrassegnato da un numero, ma cinque di essi comportano la cancellazione o la modifica dell'intero programma e vanno pertanto utilizzati con prudenza.

Un intero programma può essere cancellato

- Spegnendo il computer
- Battendo NEW
- Caricando un nuovo programma da cassetta o da disco.
- Premendo il tasto RESET.
- Battendo SYS 32768 RETURN .

Un singolo comando può essere eliminato.

- Battendone solo il numero di etichetta
- Battendo un altro comando con lo stesso numero di etichetta.

Un gruppo di comandi può essere cancellato battendo DELETE seguito dai numeri di identificazione della prima e dell'ultima riga che si vuole cancellare. Ad esempio:

DELETE 15Ø - 23Ø

cancellerà la riga 15Ø, 23Ø e tutte le righe intermedie. Il comando DELETE presenta delle varianti simili a quelle del comando LIST.

DELETE – 1ØØ	eliminerà tutte le righe fino a 1ØØ
DELETE 12Ø	eliminerà tutte le righe dalla numero 12Ø alla fine del programma
DELETE 2ØØ	eliminerà solo la riga 2ØØ
DELETE	usato da solo non cancella l'intero programma ma, viene invece consideratò un SYNTAX ERROR (errore di sintassi).

Prestare attenzione quando si usa il comando DELETE, in quanto un solo errore può causare gravi danni al programma!

Ricaricare il programma SENTENCES e cancellare alcune righe che contengono la parola chiave REM.

Controllare che la cancellazione sia stata correttamente eseguita utilizzando il comando LIST per visualizzare la porzione di programma interessata.

Aggiunta di nuovi comandi

Un nuovo comando può essere aggiunto ad un programma esistente battendolo con un adatto numero di etichetta. Il comando è inserito nel punto determinato dal numero di etichetta.

L'operazione di inserimento comandi, già utilizzata nella Lezione 5, verrà ora impiegata per inserire alcuni comandi REM. Assicurarsi di non sostituire un'istruzione esistente, altrimenti il programma non funzionerà.

Modifica di comandi esistenti

Il modo più drastico per modificare un comando consiste nel ribatterlo, usando lo stesso numero di etichetta.

Iniziare sostituendo la riga del copyright (etichetta 5) con una riga contenente il proprio nome. Un esempio potrebbe essere il seguente:

```
        ┌─►LIST 5
        │
        │  5 REM COPYRIGHT© ANDREW
        │    COLIN 1984
        │
Battere─┼─►5 REM MARIO ROSSI
        │
        └─►LIST 5

           5 REM MARIO ROSSI
```

Effettuare altre modifiche limitandosi alle righe contenenti comandi REM, altrimenti si corre il rischio di danneggiare il programma impedendogli di funzionare correttamente. Un programma è come una cellula vivente: le mutazioni casuali sono quasi sempre critiche e solitamente fatali.

Quando una riga ha bisogno di una piccola modifica, è spesso più facile variare l'originale (che è già sullo schermo) che non batterne una nuova versione. Ciò si ottiene utilizzando il cur-

sore ed eventualmente il tasto INST DEL .
Quando le modifiche sono completate, il tasto RETURN fa sì che il computer registri il nuovo comando in luogo del vecchio.

Supponiamo di voler alterare la riga 1ØØ in modo che appaia come segue:

1ØØ REM MAIN STUPID SENTENCE
GENERATOR

Listare (LIST) il comando 1ØØ e spostare il cursore sulla S di SENTENCE e inserire 7 spazi vuoti (usare i tasti SHIFT e INST DEL).
Quindi battere la parola STUPID, controllare che tutto sia corretto e battere RETURN .
Eseguire un altro LIST 1ØØ come controllo.

Provare altre modifiche di questo tipo, sempre limitandosi ai comandi REM. Notare che se non si batte RETURN dopo aver modificato una riga, la macchina non registra alcuna variazione. A questo punto battere RUN. Se il programma non funziona più, si è certamente commesso un errore nella correzione, ad esempio cancellando o modificando inavvertitamente un'istruzione. Ciò non comporta alcun inconveniente. Basta ricaricare il programma da cassetta o da disco.

Si sarà osservato che il programma SENTENCES fa delle dichiarazioni su personaggi ben noti. Le scelte possibili sono abbastanza limitate e si trovano nei comandi 9Ø7Ø (per gli uomini) e 91ØØ (per le donne). Per la parte finale di questo esercizio si procederà a modificare queste liste, in modo che il programma crei delle frasi sulla propria famiglia e sui propri amici.

Ciascuno dei due comandi 9Ø7Ø e 91ØØ contiene la parola chiave DATA. Questa è seguita da un lista di nomi, separati da virgole. L'ultimo nome è seguito da una virgola e dalla lettera Z*.

La lunghezza di queste liste di nomi può essere determinata a piacere. Tuttavia, se occupa uno spazio superiore alle due righe, è necessario usare un secondo comando DATA (con un numero di etichetta maggiore di 1 rispetto al precedente). Possono anche essere usati un terzo e un quarto comando DATA ma soltanto l'ultimo comando DATA in ciascun gruppo richiede la Z finale.

* *L'uso di Z al termine del comando DATA è una caratteristica specifica di questo programma, non del BASIC in generale. I comandi DATA nella maggior parte degli altri programmi, non richiedono la Z finale.*

Alcune possibili alternative per le righe 9Ø7Ø e 91ØØ potrebbe essere:

9Ø7Ø DATABILL,GEOFFREY,PERCEVAL, MR.SOPHOCLES,THE HEADMASTER,Z
91ØØ DATAGRANNY,SUSAN,VIOLET,MRS.PINKERTON,THE GYM MISTRESS, AUNTIE FLO, RACHEL
91Ø1 DATAPENNY,KATE,LAURA, FRANCES, NORAH, VICKY,Z

Una volta effettuate queste modifiche, eseguire nuovamente il programma. Se il risultato è completamente privo di senso, controllare la presenza di una virgola (ma non due) tra un nome e l'altro, e che l'ultimo nome sia seguito da una Z.

Dopo aver acquisito familiarità con le operazioni di modifica sarà possibile esercitarsi a variare le liste di parole contenute nel programma:

9ØØØ Azioni che compiono autonomamente (verbi intransitivi)
9Ø1Ø Azioni che si compiono con altre persone (verbi transitivi)
9Ø2Ø Azioni riguardanti l'abbigliamento (verbi transitivi)
9Ø3Ø Indumenti maschili
9Ø4Ø Indumenti femminili
9Ø5Ø Lista di avverbi e frasi avverbiali descriventi azioni che si compiono con altre persone.
9Ø6Ø Lista di avverbi e frasi avverbiali descriventi azioni che compiono autonomamente.
9Ø7Ø Nomi di uomo
9Ø8Ø Aggettivi relativi a uomini
9Ø9Ø Vari tipi di uomo
91ØØ Nomi di donna
911Ø Aggettivi relativi a donne
912Ø Vari tipi di donna

Modifica le liste in vari modi, facendo attenzione a mantenerle coerenti. Ad esempio, se si altera la riga 9Ø2Ø, facendo in modo che contenga azioni riguardanti il cibo, si devono di conseguenza alterare le righe 9Ø3Ø e 9Ø4Ø, altrimenti si potrebbero avere frasi di questo genere

SUSAN ATE HER WELLINGTON BOOTS

(Susan ha mangiato i suoi stivali Wellington)

Esercizio 6.2 completato	

ESERCIZIO 6·3

Una volta modificato il programma SENTENCES, facendo in modo che compaiano sullo schermo frasi divertenti sui propri amici, è possibile conservare la nuova versione per mostrarla in occasione di feste, ecc. Questa sezione spiega come conservare il programma su una cassetta. Quanto segue non riguarda gli utenti in possesso di unità a disco.

Procurarsi un nastro vuoto o contenente informazioni che non si desidera conservare. Deve trattarsi di un nastro di buona qualità ed essere il più breve possibile: si userà infatti soltanto circa 1 minuto della durata del nastro e non vale la pena di spendere di più per una durata maggiore. Le cassette della Commodore sono l'ideale.

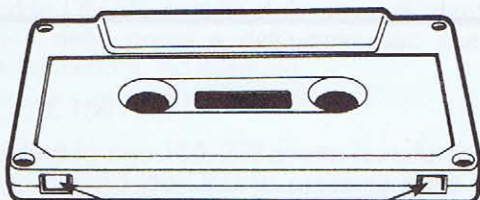
Caricare il nuovo nastro nell'unità a cassetta in luogo della cassetta SENTENCES e riavvolgerlo. Sbloccare tutti i tasti sull'unità a cassetta, quindi interrompere il programma SENTENCES e battere

SAVE"FAMIGLIA" RETURN

(è possibile usare qualsiasi nome invece di FAMIGLIA).
La macchina risponde

PRESS RECORD AND PLAY ON TAPE
(premere RECORD e PLAY)

Seguire queste istruzioni premendo entrambi i tasti sul registratore contemporaneamente.

Se il tasto RECORD non si abbassa, controllare la cassetta per assicurarsi che non siano state tolte le linguette di protezione scrittura. Queste linguette si trovano nella parte posteriore della cassetta come indicato in figura:

il loro scopo è di proteggere le registrazioni importanti che non devono essere distrutte. Se le linguette sono state asportate, è impossibile registrare, per cui in questo caso occorre procurarsi un altro nastro.

Se tutto procede regolarmente, verrà visualizzato

SAVING FAMIGLIA

e in un momento successivo

READY.

In teoria il programma è ora registrato, ma vale la pena di controllare dal momento che potrebbero essersi verificati vari inconvenienti: è possibile aver dimenticato di riavvolgere il nastro o di premere il pulsante RECORD, oppure il nastro stesso potrebbe avere un piccolo difetto che gli impedisce di eseguire una copia corretta del programma. Queste cose non dovrebbero succedere, ma in pratica succedono!

Per controllare il nastro, riavvolgerlo, quindi battere

VERIFY"FAMIGLIA" RETURN

Il computer risponde

PRESS PLAY ON TAPE

Premere il pulsante PLAY (questa volta senza il pulsante RECORD). A questo punto il computer cerca il programma su nastro e lo controlla a fronte di ciò che c'è nella memoria. Naturalmente non si devono fare modifiche tra i comandi SAVE e VERIFY.

Se tutto procede correttamente, i messaggi che si vedranno comparire saranno:

```
VERIFY "FAMIGLIA"
PRESS PLAY ON TAPE
OK
SEARCHING FOR FAMIGLIA
FOUND FAMIGLIA
VERIFYING
OK
```

Se viene riscontrato un errore, oppure se l'operazione si arresta prima di giungere al messaggio FOUND FAMIGLIA, occorre ripartire da capo e con il comando SAVE.

Se il difetto persiste, provare un altro nastro (oppure l'altro lato dello stesso nastro). Se non è possibile ancora far funzionare il sistema, portare il computer e la cassetta al rivenditore per un controllo.

Una volta che il programma è stato salvato (SAVE), può essere riposto altrove e caricato (LOAD) in qualsiasi momento, con un comando tipo

LOAD"FAMIGLIA" RETURN

Per salvare un programma su disco rimuovere il disco ITB PROGRAMS e caricare quello formattato all'inizio della lezione. Registrare ora la propria versione del programma con il comando

DSAVE"nome programma" RETURN

dove "nome programma" è il titolo che è stato scelto per il programma. Ad esempio

DSAVE"FAMIGLIA" RETURN

Se tutto procede regolarmente, viene visualizzato il messaggio:

SAVING 0:FAMIGLIA

e un attimo dopo

READY.

Per assicurarsi che il programma sia stato registrato correttamente, battere VERIFY"nome programma",8 RETURN (dove "nome programma" è il nome attribuito a scelta). Notare che nel comando VERIFY (al contrario di DSAVE) il nome del programma deve essere seguito dalla sequenza ,8. Verrà visualizzato a questo punto

SEARCHING FOR nome programma
VERIFYING
OK
READY.

Questo processo funziona sempre praticamente come è descritto, in caso contrario, riprovare da capo attentamente, fino alla fine. Se non riesce ancora, chiedere consiglio al rivenditore.

Una volta salvato (DSAVE) il programma ed averlo verificato (VERIFY), caricarlo e listare il catalogo in cui dovrebbe ora comparire il programma appena caricato. Un metodo più veloce per battere la sequenza DSAVE" è premere il tasto f2 f5 contemporaneamente a SHIFT.

Una volta che un programma è stato salvato può essere ricaricato nel modo solito da un comando del tipo

DLOAD"FAMIGLIA" RETURN

Disponendo di un disco formattato, è possibile continuare ad inserirvi programmi fino al termine dello spazio disponibile. Di tanto in tanto può essere necessario eliminare un programma e sostituirlo con una nuova versione avente lo stesso nome. Se si impartisce il comando

DSAVE"nome programma"

e sul disco esiste già un programma con quel nome, l'unico risultato è il lampeggiamento della spia rossa. Per eliminare il vecchio programma occorre aggiungere un carattere @ davanti al nome del programma.

Il comando si presenta ora come segue:

DSAVE"@ nome programma"

È opportuno far seguire questo tipo di comando DSAVE dal comando speciale

OPEN 1,8,15,"V"

e attendere che la spia rossa si spenga. Ciò permette al computer di risparmiare memoria, raggruppando gli spazi vuoti che si sono creati eliminando il vecchio programma.

Un programma non deve essere perfetto per poter essere salvato.

Se si scrive un programma molto lungo (o addirittura se se ne copia uno da un libro), è consigliabile eseguire un comando SAVE ogni mezz'ora circa. Ciò in quanto la memoria del computer non è affidabile quanto un dischetto o un nastro riposti in un cassetto.

È improbabile che il computer si guasti, ma si possono verificare altri incidenti. Può capitare ad esempio che un fulmine danneggi le informazioni contenute in memoria, potrebbe verificarsi una mancanza di corrente, oppure qualcuno potrebbe camminare sul cavo di alimentazione e staccarlo dalla presa. Questo è poco piacevole dopo che si sono spese ore e ore di lavoro sul computer. Se, per contro, sono state effettuate regolari copie della memoria ogni mezz'ora, è possibile ricaricare la versione più recente e procedere soltanto con una piccola perdita di tempo.

Per rendere il sistema assolutamente sicuro, sarebbe opportuno eseguire le operazioni di salvataggio su due diversi nastri o dischi, alternativamente. Ciò assicura la protezione anche nel caso in cui il computer si fermi durante una operazione SAVE, con la metà della vecchia versione coperta da metà della nuova.

Esercizio 6.3 completato	

Si cercherà ora di chiarire un punto che può generare errori di cui non è facile scoprire la causa; verranno inoltre dati suggerimenti ai programmatori del computer Commodore sul modo di uscire da questa eventuale "trappola".

Per cominciare si cercherà di metterne per iscritto un semplice esempio. Battere NEW per cancellare la memoria e quindi immettere il seguente programma inserendo accuratamente tutti gli spazi indicati e osservando lo schermo mentre si batte.

```
1Ø PRINT"A FRIGHTFUL AND APPALLING TRAP"
2Ø GOTO 1Ø
```

Ora battere il comando LIST e controllare il programma, che dovrebbe comparire esattamente come indicato.

A questo punto ci si aspetterebbe che battendo RUN apparisse ripetutamente il messaggio

A FRIGHTFUL AND APPALLING TRAP

fino a che il programma non viene fermato. Controllare il risultato. È probabile invece che compaia

A FRIGHTFUL AND APPALLING TRAP 2Ø
?SINTAX ERROR IN 1Ø
READY.

Anche se il programma funziona correttamente occorre leggerlo per scoprire come si è fatto per evitare la trappola.

Il motivo per cui la macchina non è stata in grado di eseguire il programma (ammesso che sia così), non è facile da scoprire a prima vista. Anche un esperto di programmazione BASIC avrebbe delle difficoltà a localizzare l'errore.

La difficoltà sorge a causa della larghezza dello schermo del computer. All'interno della macchina, qualsiasi comando BASIC può avere una lunghezza massima di 75 caratteri. Lo schermo è largo soltanto 4Ø caratteri cosicchè la versione *visualizzata* di un comando può distribuirsi su un massimo di 2 righe di schermo.

Quando si batte un comando e il cursore raggiunge la fine di una riga dello schermo, il sistema lo sposta all'inizio della riga successiva, ma presume ancora che stia battendo lo stesso comando. Un comando viene terminato soltanto dal tasto RETURN .

Il nostro esempio presuppone che la trappola sia "scattata" ed ecco come ciò avviene:

si batte il primo comando (che è stato opportunamente studiato per riempire l'intera riga dello schermo). A qusto punto il cursore si trova all'inizio della riga successiva e naturalmente si batte il comando successivo, terminandolo con un RETURN . Dato che non si è terminata la prima riga con un RETURN , il sistema interpreta le due righe seguenti come parte dello stesso comando.

1Ø PRINT"A FRIGHTFUL AND APPALLING TRAP" 2Ø GOTO 1Ø

Questo "comando" non è scritto correttamente, secondo le regole del BASIC e dà luogo a un errore di sintassi quando la macchina cerca di eseguirlo.

Questo tipo di errore è particolarmene difficile da trovare, a meno che non si sappia che cosa si sta cercando. È molto improbabile notare l'errore quando si batte il programma – anche i programmatori esperti spesso dimenticano di terminare i comandi con RETURN se il cursore si trova all'inizio di una nuova riga. Se si lista il programma o anche solo la sezione che comprende l'errore, il comando difettoso assomiglia esattamente a due comandi corretti e il difetto è invisibile.

Fortunatamente l'errore può essere individuato listando il comando in cui è segnalato l'errore. Se si batte LIST 1Ø, escono le righe 1Ø e – apparentemente – 2Ø! Ciò deve essere sbagliato dato che si è chiesta soltanto la riga 1Ø. Per correggere l'errore, ribattere entrambi i comandi completamente, ricordando di terminare ciascuno di essi con RETURN

Riassumendo:

(a) Terminare sempre qualsiasi comando con RETURN , ovunque si trovi il cursore.

(b) Se il computer segnala un errore in un comando e non si riesce a riscontrare alcun errore, listare isolatamente il comando interessato e controllare se interferisce con il comando successivo.

Esercizio 6.4 completato	

LEZIONE: 7

I programmi scritti per esercizio nella Lezione 5 erano privi di qualsiasi controllo. Una volta avviati, per interromperli occorre un'azione drastica, in assenza della quale l'esecuzione sarebbe stata teoricamente infinita. Questa Lezione tratta del modo in cui controllare i programmi e interromperli dopo un certo periodo di tempo.

Gli argomenti descritti in questa Lezione sono fondamentali per la programmazione e una volta che si siano compresi a fondo si sarà compiuto il passo più importante per diventare programmatori. Leggere la lezione lentamente e attentamente, e se rimangono dei dubbi su qualche punto, tornare indietro e leggerla di nuovo. Vale la pena di compiere uno sforzo in più data l'importanza dell'argomento trattato.

Il controllo dei programmi dipende da un concetto chiave che per qualcuno potrebbe essere nuovo: la *condizione*.

Nell'ambito di una normale conversazione, la maggior parte delle affermazioni che si fanno sono vere, o per lo meno vengono prese come tali fino a prova contraria, ad esempio:

"Il treno su cui viaggio ha avuto un guasto"

"Ti amo"

Una condizione è un tipo speciale di dichiarazione che non è necessariamente vera ma che potrebbe ugualmente anche essere falsa. In inglese, si usano le condizioni dopo la parola "if" (se). Nelle seguenti frasi, le condizioni sono evidenziate in neretto:

"Se **l'ultimo treno è partito,** occorrerà trascorrere la notte a Milano".

"Se **il programma non funziona,** occorre trovare l'errore e correggerlo."

Chi pronuncia queste frasi non sta insistendo sul fatto che il treno sia partito o che il programma realmente non funzioni; egli semplicemente non lo sa e sta preparando dei piani di conseguenza. Una condizione può risultare vera o falsa senza che per questo chi la esprime sia necessariamente un bugiardo.

Anche nel BASIC le condizioni sono espresse dopo la parola chiave IF. Esse coinvolgono i vari "elementi" usati nei programmi: variabili numeriche, variabili stringa, numeri e stringhe. Le condizioni, che in ogni caso possono essere vere o false, sono costruite utilizzando una delle sei relazioni possibili. Ciò è meglio illustrato dall'esempio:

Si consideri la condizione BASIC:

A<5

(dove < è un segno che significa "minore di"). Questa condizione è vera se il valore della variabile A è realmente minore di 5 (ad esempio Ø o 3 o 4.98). È *falsa* se A vale 5 o più di 5.

Un altro esempio stavolta con l'uso delle stringhe:

N$<>"JIM"

(dove <> significa "è diverso da")

Questa condizione è vera se la variabile N$ ha qualsiasi valore salvo "JIM"; perciò è vera se N$="JACK" o N$="JIMMY". È falsa solo se N$ è effettivamente "JIM".

La serie completa di relazioni che è possibile usare nel BASIC comprende:

= (uguale a)
< (minore di)
> (maggiore di)
<> (diverso da)
<= (minore o uguale a)
>= (maggiore o uguale a)

Le relazioni <>, <= e >= richiedono l'uso di due tasti ciascuna. Questi simboli possono essere più familiari nelle forme ≠ N, e ≥ ma i progettisti del BASIC hanno dovuto accettare il fatto che le tastiere dei computer non prevedono solitamente questi segni.

Le relazioni possono essere tutte usate sia tra coppie di numeri che tra coppie di stringhe per porre delle *condizioni*. I numeri e le stringhe possono essere rappresentati da variabili appropriate.

5>4 è *vero* in quanto 5 è maggiore di 4
7<=6 è *falso* in quanto 7 è maggiore di 6
se A = 1Ø e B =7,
A>=B è *vero* e così B<=7.

Quando vengono usate relazioni tra le stringhe, queste implicano l'ordine alfabetico (come da dizionario) in modo che

"CANE"<"GATTO" è vero,

e

"GIOVANNI">"GIOVANNINO" è falso.

ESERCIZIO 7.1

Supponiamo che il computer abbia eseguito le seguenti tre istruzioni:

```
LET A$="JOAN"
LET X=5
LET Y=7
```

Elaborare la tabella seguente e contrassegnare ciascuna condizione come falsa o vera:

Condizione	Valore (vero o falso)
X < 7	
X >= 5	
A$<>"X"	
Y <> X	
A$ < "FRANCES"	
A$ > "JOAN"	
Y = 8	

Le quantità su entrambi i lati della relazione possono essere espressioni, esattamente come nei comandi LET. Le espressioni possono essere complesse a piacere, ma la cosa importante è il confronto fra uguali: una condizione che ha un numero su un lato ed una stringa sull'altro fa sì che il computer si fermi e segnali un errore. Elaborare le seguenti condizioni supponendo che i valori di A$, X e Y siano gli stessi dell'esempio precedente:

Condizione	Valore (vero o falso)
A$+"NE"<>"JOANNE"	
5 > X	
X + Y <> 13	
X + 2 = Y	

Controllare ora le risposte, che sono indicate nell'Appendice B.

Esercizio 7.1 completato	

Lo strumento principale di controllo in BASIC è il comando IF. Esso si compone di una parola chiave IF, una condizione, la parola THEN e un numero di etichetta. È molto simile a GOTO, ma con una differenza: il salto avviene soltanto se la condizione è *vera*.

Un esempio del comando IF è

IF X$<>"ABBBB"THEN 2Ø

Qui la condizione è X$ <>"ABBBB" e l'intero comando impone al computer di saltare all'istruzione 2Ø se X$ è diverso da "ABBBB". Se questa condizione è falsa, la macchina continua ad eseguire i comandi nel loro ordine numerico.

All'inizio, la maggior parte delle persone rimane perplessa di fronte a questo comando, in quanto la variabile X$ può essere uguale o diversa dalla stringa contenente la A e la B. La scelta tra questi due casi dipende dai comandi precedenti, che dovrebbero comunque essere ben noti al programmatore che ha inserito il comando IF.

Questo punto di vista è comprensibile ma sbagliato, dal momento che:

- il comando IF può appartenere ad un loop in cui una certa variabile modifica il proprio valore ad ogni passaggio.
  La condizione potrebbe essere anche vera per alcuni dei valori ma non per altri.
- oppure è possibile che si stia scrivendo un programma che deve essere usato da un'altra persona.

  In tal caso occorre prevedere tutte le combinazioni che possono derivare dall'uso del programma.

  Ad esempio: occorre fare in modo che i diversi programmi di quiz rispondano in modo coerente alle risposte, anche se queste non sono conosciute in anticipo.

L'inserimento di un'istruzione IF in un'iterazione fornisce un modo interessante per interromperla dopo averla ripetuta un certo numero di volte. Battere ed eseguire quanto segue:

```
1Ø X$="A"
2Ø PRINT X$
3Ø X$=X$+"B"
4Ø GOTO 2Ø
5Ø STOP
```

Questo programma viene eseguito riempiendo lo schermo con stringhe di B sempre più lunghe fino a che non si esaurisce tutto lo spazio. Il comando STOP alla riga 5Ø non viene mai raggiunto.

Ora, fermare il programma e sostituire la riga con 4Ø con

```
4Ø IF X$<>"ABBBB"THEN 2Ø
```

Quando si esegue il programma, sullo schermo compare

```
A
AB
ABB
ABBB
```

e poi il programma si ferma!

Il motivo sta nella condizione X$<>"ABBBB". Quando il programma ripete l'iterazione, la condizione è all'inizio vera (in quanto X$ assume di volta in volta i valori AB, ABB e ABBB, che sono tutti diversi da ABBBB).

In ciascuno di questi casi, il comando IF si comporta come un semplice GOTO 2Ø, facendo in modo che venga eseguita un'altra iterazione. Alla fine, X$ assume il valore ABBBB. La condizione è ora falsa: il salto non si verifica, e la macchina esegue l'ultima istruzione (5Ø STOP), dopodiché si arresta.

Ora provare a cambiare la condizione in vari modi ed osservare l'effetto durante l'esecuzione del programma. Qualsiasi stringa si usi, assicurarsi che ad un certo punto la condizione risulti falsa, altrimenti il programma non si fermerà mai.

Le possibili condizioni da provare sono:

X$<>"AB"
X$<>"ABBBBBBBBBBB"
X$<"ABBA"

Lo stessa tecnica di cotrollo può essere usata con le variabili numeriche.

Battere

```
1Ø P=Ø
2Ø PRINT P,P★P
3Ø P=P+1
4Ø GOTO 2Ø
5Ø STOP
```

Eseguire questo programma e controllare il risultato, fermarlo e cambiare la riga 4Ø in modo che appaia come segue:

```
4Ø IF P<11 THEN 2Ø
```

Eseguire nuovamente il programma. Esso visualizza due colonne di cifre che appaiono familiari e che potrebbero essere utili a chi non conosce a memoria i quadrati dei numeri.

Dal punto di vista funzionale si può evidenziare un difetto: la visualizzazione non contiene alcuna dicitura (le label) e quindi il suo significato non è immediatamente comprensibile, se non da parte di chi ha scritto il programma. È possibile rimediare aggiungendo all'inizio un titolo o una riga che identifica ciascuna colonna come segue:

```
NUMERO          QUADRATO
  0                 0
  1                 1
  2                 4
  3                 9
  . .           . . . . .
ecc.
```

Chiaramente il titolo deve essere visualizzato prima di qualsiasi numero o del relativo quadrato per cui il comando che lo visualizza deve venire per primo. Dato che è già stata usata l'etichetta 1Ø è che sarebbe troppo laborioso cambiare tutte le etichette dell'intero programma, una soluzione ragionevole potrebbe essere quella di utilizzare l'etichetta 5. Il comando è di per sè un PRINT con due stringhe: "NUMERO" e "QUADRATO". Le virgole tra le stringhe assicurano che la spaziatura corrisponda a quella esistente tra le colonne di cifre. L'intero programma risulta ora come segue:

```
 5 PRINT "NUMERO","QUADRATO"
1Ø P=Ø
2Ø PRINT P,P★P
3Ø P=P+1
4Ø IF P<11 THEN 2Ø
5Ø STOP
```

Eseguire il programma in questa forma ed esaminare il risultato.

Se si desidera una riga vuota tra il titolo e la prima fila di cifre, occorre utilizzare un comando PRINT da solo (cioè non seguito da alcun valore o stringa), per ottenere una riga vuota. Provare ad aggiungere il comando

```
7 PRINT
```

Alla fine di questa Lezione sarà possibile iniziare a progettare alcuni semplici programmi. Prima di iniziare a fare ciò, occorre tuttavia dare un sguardo attento ai programmi già eseguiti e ricavare alcune conclusioni generali. I programmi esemplificativi sono:

(1)

```
1Ø X$="A"
2Ø PRINT X$
3Ø X$=X$+"B"
4Ø IF X$ <>"ABBBB" THEN 2Ø
5Ø STOP
```

(2)

```
 5 PRINT"NUMERO","QUADRATO"
 7 PRINT
1Ø P=Ø
2Ø PRINT P,P★P
3Ø P=P+1
4Ø IF P < 11 THEN 2Ø
5Ø STOP
```

Tralasciando i comandi relativi al titolo (5 e 7) contenuti nel secondo programma, i due programmi presentano una struttura simile. Infatti in entrambi i casi è presente:

1. una variabile (X$ nel primo programma e P nel secondo) che cambia regolarmente, mano a mano che l'iterazione viene ripetuta. In generale, questa è chiamata una variabile di *controllo* è può essere una stringa o un numero.
2. un comando che dà alla variabile di controllo il suo valore *iniziale*. Questo comando è al di fuori dell'iterazione (cioè non viene ripetuto ma viene eseguito soltanto una volta)
3. un comando che viene eseguito per ogni valore della variabile di controllo. Negli esempi, questi sono i comandi PRINT.

   PRINT X$
   e PRINT P,P★P

   In pratica, questa parte dell'iterazione può essere ampliata per includere qualsiasi numero di comandi, tutti eseguiti per ciascun valore della variabile di controllo. Questo gruppo è detto *corpo* dell'iterazione.
4. un *incremento* o quantità di cui la variabile di controllo cresce ogni volta che viene ripetuta l'iterazione. Nei nostri esempi, X$ cresce aggiungendo una "B" e P viene aumentato di 1. Sono possibili altri incrementi; per esempio, una stringa potrebbe crescere di 5 simboli alla volta o un numero potrebbe aumentare a incrementi di 2 (o di qualsiasi altro valore). Potrebbe anche iniziare con un valore alto e scendere o "decrementare" contando alla rovescia. L'iterazione comprende sempre un comando che sposta la variabile di controllo di un passo ogni qualvolta viene eseguito.

5. un valore finale per la variabile di controllo. Quando l'iterazione viene eseguita con questo valore, la ripetizione deve cessare. L'ultimo comando nell'iterazione è un comando IF, con una condizione che è vera se l'iterazione deve essere ancora eseguita, ma falsa quando la variabile di controllo ha superato il suo valore finale.

Nella tabella che segue, si esamini ciascun programma e s'inserisca il nome della variabile di controllo, il valore iniziale, il valore finale, l'incremento e il numero delle iterazioni eseguite. Per fare ciò è spesso di aiuto annotare i valori assunti dalla variabile di controllo ogni volta che l'iterazione viene eseguita (1ª, 2ª, 3ª volta, ecc.) e vedere quanti valori ci sono fino al raggiungimento del valore finale.

	Variabile di controllo	Valore iniziale	Valore finale	Incremento	N. dei passaggi del ciclo
1Ø X$="A" 2Ø PRINT X$ 3Ø X$=X$+"B" 4Ø IF X$<>"ABBBB"THEN 2Ø 5Ø STOP	X$	"A"	"ABBB"	"B"	4
1Ø P=Ø 2Ø PRINT P,P*P 3Ø P=P+P 4Ø IF P < 11 THEN 2Ø 5Ø STOP	P	Ø	1Ø	+1	11
1Ø Y$="Z" 2Ø PRINT Y$ 3Ø Y$=Y$+"XY" 4Ø IF Y$ <>"ZXYXYXY"THEN 2Ø 5Ø STOP					
1Ø R=5 2Ø PRINT R, R/8 3Ø R=R+3 4Ø IF R < 17 THEN 2Ø 5Ø STOP					
1Ø C=27 2Ø H=3Ø−C 3Ø PRINT C,H 4Ø C=C−5 5Ø IF C > 2 THEN 2Ø 6Ø STOP					

Una volta completata la tabella, controllare le risposte a fronte di quelle indicate al termine del manuale (Appendice B).

Esercizio 7.2 completato	

ESERCIZIO 7·3

Quando si progetta un programma, occorre iniziare pianificandolo in qualche modo e quindi scrivendo l'intero programma su un pezzo di carta. Usare gomma e matita! Alcuni compongono i programmi direttamente sulla tastiera del computer, ma questo metodo è adatto soltanto per i geni e per gli sciocchi – è chiaramente non consigliato per le persone normali. La ragione è abbastanza semplice: se s'inizia senza avere le idee chiare su ciò che si deve fare, si hanno altrettante probabilità di successo di un costruttore che imposta una casa senza disegni, creando l'architettura mano a mano che procede. Quel costruttore potrebbe anche produrre un gioiello architettonico, ma più probabilmente terminerà con una stamberga che cadrà a pezzi al primo temporale.

Quando si progetta un loop per un programma, occorre decidere tutti gli elementi essenziali.

Questi comprendono il tipo e il nome della variabile di controllo; i valori inziali e finali, l'incremento e i dettagli per il corpo del loop. Una volta chiariti questi punti, è possibile dedurre un sistema standard di progettazione.

Ecco un esempio pratico.

Una sterlina vale all'incirca 2350 lire italiane. Si tratta di costruire una tabella che dia l'equivalente italiano degli importi da 5 a 75 sterline, procedendo a incrementi di 5 sterline.

E cioè:

Lire sterline	Lire italiane
5	11750
10	23500
. . . .	. . . .

e così via

Pensiamo per prima cosa all'iterazione. La variabile di controllo chiaramente sarà un nu-

mero che si potrebbe chiamare PS (che sta per Pounds Sterling - Lire sterline). Il valore iniziale sarà 5 e il valore finale 75 e l'incremento 5. Il corpo dell'iterazione deve stampare un valore in Lire sterline e il corrispondente valore in lire che è 2350 volte tanto.

Si possono quindi scrivere gli elementi dell'iterazione, che sono:

```
PS=5                  (Definisce il valore iniziale)
PRINT PS,2350★PS                             (Corpo)
PS=PS+5                              (Incrementa PS)
IF PS<80THEN      (Controlla se il valore finale
                                       è superato)
STOP                    (Interrompe il programma)
```


Il numero di etichetta che segue THEN viene lasciato in bianco, dato che per il momento non è noto.

Prima di scrivere l'intero programma, occorre considerare l'intestazione. Adatti comandi potrebbero essere:

PRINT"£","LIRE"

e PRINT (per ottenere una riga vuota)

È ora possibile assemblare le varie parti e scrivere l'intero programma:

```
10 PRINT"£","LIRE"
20 PRINT
30 PS=5
40 PRINT PS,2350★PS
50 PS=PS+5
60 IF PS<80 THEN 40
70 STOP
```

A rischio di risultare noiosi, è necessario ripetere: non cercare di abbreviare o di èvitare la progettazione iniziale; non improvvisare il programma direttamente sul computer, altrimenti non si diventerà mai programmatori.

Provare ora questi esempi:

1. Scrivere un programma che visualizza un profilo di asterischi, cioè:

   ★
   ★★
   ★★★
   ★★★★
   ....
   fino a
   ★★★★★★★★★★

2. Scrivere un programma che dia l'equivalente in dollari americani per somme in moneta britannica comprese tra 1Ø e 3Ø sterline, procedendo a incrementi di 2 sterline (1 sterlina =1,43 $).

3. La relazione tra le scale dei gradi centigradi e Fahrenheit è espressa da questa formula:

F=1.8★C+32

Scrivere un programma che stampi le equivalenti temperature in Fahrenheit delle temperature in gradi centigradi comprese tra 15 e 30°C, procedendo a intervalli di 1°C.

(SUGGERIMENTO: il corpo del programma potrebbe essere

F=1.8★ C+32

PRINT C,F

Ciò è importante per la scelta del nome della variabile di controllo).

Una volta scritti ed eseguiti tutti questi programmi, controllare le soluzioni a fronte di quelle contenute nell'Appendice B.

Esercizio 7.3 completato	

Coloro che amano la matematica, talvolta vengono confusi dal modo in cui viene usato il segno "=" in BASIC.

In matematica "=" è usato nelle equazioni per asserire che due espressioni diverse hanno in effetti lo stesso valore. L'espressione esprime cioè una condizione vera. Per esempio, se l'insegnante di matematica scrive sulla lavagna

"2x+5=9"

è possibile essere sicuri che per quel particolare x, che l'insegnante ha in mente, l'affermazione è corretta. Se così non fosse, sarebbe possibile immaginare la seguente conversazione:

L'allievo alza la mano:

Insegnante:	Sì
Allievo:	x è due
Insegnante:	No. La risposta è 78
Allievo:	Eh? Non capisco.
Insegnante:	Ho detto una bugia scrivendo 2x+5=9!

In BASIC "=" è usato in due sensi diversi, nessuno dei quali corrisponde a quello usato in matematica.

In un comando LET, il segno significa "diventa". Si tratta di un'istruzione che calcola il valore dell'espressione a destra dell'uguale e lo attribuisce alla variabile posta sulla sinistra. Le istruzioni non sono affermazioni e non ha quindi senso indagare se siano vere o meno (potrebbero esserlo o non esserlo, ma ciò è qui irrilevante).

Il problema è che se viene trascurata la parola chiave LET, il comando assomiglia a un'equazione, pur non essendolo. Chiariamo questo punto:

in BASIC

Y=X+2

non informa il computer che Y è uguale a X+2 ma gli ordina di calcolare il valore di X+2 e di inserire il risultato nella variabile Y. Ecco alcuni punti da considerare:

- Q=Q+5 è un comando BASIC utile e ragionevole
- P=Q e Q=P non hanno gli stessi effetti
- X+1=5 non è un comando BASIC lecito

Dopo aver riflettuto sul significato di questi punti cercare di esprimerli con parole proprie.

L'altro uso di "=" è nelle condizioni. Si ricorderà che = è una delle sei possibilità di relazione tra quantità.

Gli esempi del suo uso sono

```
IF X=Y+2 THEN 1ØØ
IF N$="YES" THEN 15Ø
```

Anche in questo caso non è implicito che la condizione sia vera; al contrario, il comando ordina al computer di verificare se la condizione è vera e di intraprendere una determinata azione nel caso in cui lo sia.

Nelle condizioni, "=" ha la stessa forza logica di qualsiasi altra relazione tipo < o >=.

È quindi meglio evitare l'espressione "uguale a" e chiamare il simbolo " è lo stesso di".

Per riassumere:

il BASIC usa "=" nei comandi LET, dove significa "diventa" e nelle condizioni dove significa "è lo stesso di", ma l'affermazione espressa dal segno "=" non è necessariamente vera.

Il programma di auto-test per questa lezione è detto UNIT7QUIZ.

LEZIONE: 8

A questo punto del corso inizieremo a scrivere alcuni programmi. I primi sono corti e semplici. Successivamente, mano a mano che si approfondiscono conoscenza, esperienza e capacità, si potranno progettare e sviluppare programmi sempre più complessi e interessanti.

La seguente tabella dà qualche idea a riguardo.

Programma	*Numero di comandi*
Conversione Lire italiane in Lire sterline (Lezione 7)	7
Programma quiz Lezione 3	ca. 100
Programma per giocare a scacchi	ca. 5000
Programma per controllare un robot industriale	ca. 25000
Programma per gestire un sistema di prenotazioni aeree computerizzato	ca. 5000000

Naturalmente qualsiasi programma con più di circa 5000 comandi è sempre il risultato di uno sforzo di gruppo (occorrerebbe troppo tempo ad una sola persona per scriverlo) ma in ogni caso c'è sempre abbastanza spazio per il singolo programmatore.

Lavorando alla programmazione, ci si troverà spesso bloccati. Un programma per quanto scritto e immesso con grande cura non sempre riesce a soddisfare le aspettative. Questa lezione descrive alcuni dei modi in cui è possibile superare le eventuali difficoltà. Occorre quindi leggerla e svolgere gli esercizi, ricordando che è consigliabile rileggere da capo ogniqualvolta sia necessario.

Si osservano varie reazioni nelle persone che incontrano le prime difficoltà di programmazione. Alcuni si sentono insultati e furiosi; altri gettano la spugna e decidono che la programmazione non è cosa che li riguardi, alcuni sostengono che il programma "è giusto al 99,9%" e passono al successivo problema ma nessuna di queste reazioni è dettata dal buon senso. La sola cosa da fare è di trovare l'errore e correggerlo. Può essere di grande conforto ricordare che ogni programmatore spesso si blocca e ciò vale anche per coloro che lavorano sui computer da 25 anni.

Gli errori di programmazione si possono classificare in tre gruppi. Il primo tipo, che è anche il più comune, è quello che si manifesta con SINTAX ERROR quando il computer tenta di eseguire un particolare comando. Ciò significa che il comando non segue le regole del linguaggio BASIC. Per esempio, potrebbero esserci un errore di ortografia in una parola chiave o potrebbero esserci virgolette in più o in meno all'interno di un programma. Gli errori di sintassi sono in gran parte provocati da errori di battitura e sono estremamente facili da identificare; l'Appendice C dà in ogni caso un elenco di controllo del tipo di errore da consultare in caso di difficoltà. Il secondo tipo di errore si ha quando il computer trova un particolare comando impossibile da eseguire. Supponiamo che la macchina arrivi al comando

130 GOTO 500

ma che non esista alcun comando con l'etichetta 500. Ciò provoca l'arresto della macchina che visualizza il seguente messaggio:

UNDEFINED STATEMENT IN 130

Sfortunatamente i messaggi di errore tendono ad essere scritti nel gergo dei programmatori, anzichè nel linguaggio normale, ma sono comunque spiegati a fondo nell'Appendice C.

Quando appare un messaggio d'errore, è utile il comando HELP. Questo comando fa sì che il computer visualizzi la riga di programma che ha causato l'errore, con la parte errata in caratteri invertiti. Per controllo, inserire e lanciare il programma:

10 PRINT 1;2:3;4
(I due punti costituiscono un errore voluto).

Il programma si interromperà dopo aver visualizzato 1 e 2.

Dopo aver dato il comando HELP apparirà:
10 PRINT 1;2:3;4

(3;4 lampeggianti) a dimostrazione che l'errore si trova vicino a 3.

La funzione HELP non è sempre molto chiara (provare ad esempio PRINT 1+2+3★★4) ma ne è tuttavia utile l'impiego.

Si può richiamare il comando HELP battendo il tasto HELP f7, oppure battendo le lettere H-E-L-P RETURN.

Il terzo tipo di errore di programmazione è il più difficile da trovare e da correggere. Esso non dà luogo a messaggi di errore, ma fa sì che il computer visualizzi la risposta sbagliata al problema o si blocchi in un loop senza addirittura visualizzare nulla. La prima e più ovvia cosa da fare è di fermare la macchina, listare il programma ed esaminarlo accuratamente. Ciò solitamente aiuta ad individuare l'errore. Supponiamo invece che non avvenga, ed immaginiamo di aver dedicato alcuni minuti ad esaminare ciascun comando senza aver trovato nulla di sbagliato.

In questi casi occorre un metodo più potente per esaminare il funzionamento del programma. Questo metodo è detto "tracciamento del

programma" e consiste nel simulare il comportamento del computer: cominciando all'inizio del programma, si procede comando per comando, fino a che non si riesce ad individuare la causa del problema. Occorre essere pazienti e metodici e soprattutto escludere momentaneamente la propria intelligenza e lavorare attraverso la serie di istruzioni, come un robot, senza cercare di formulare "ipotesi plausibili", generalizzazioni o usare qualsiasi altro tipo di scorciatoia.

Per imitare il computer occorre per prima cosa avere un'idea del modo in cui funziona. Supponiamo di poter in qualche modo "congelare" il computer tra due comandi nel mezzo dell'esecuzione di un programma, di aprirlo e osservarne l'interno, evidenziando i seguenti elementi*:

Innanzitutto il programma stesso: caricato nella memoria, più o meno nella forma in cui è stato originariamente battuto.

In secondo luogo, le varibili che il programma ha usato fino a questo punto. Ciascuna variabile occupa un certo spazio in memoria ed ha un *valore*, che potrebbe essere un numero o una stringa.

Terzo, il computer ha tenuto nota della posizione raggiunta all'interno del programma. Da qualche parte (e precisamente in una variabile speciale detta "contatore di programma") viene memorizzato il numero di etichetta del prossimo comando da eseguire.

Proviamo ora a scongelare brevemente il computer, quanto basta per eseguire un comando. Il comando che la macchina sceglie sarà naturalmente quello memorizzato dal puntatore di programma. Osservando in un momento successivo, si troveranno certamente delle modifiche che dipendono dal comando che è stato appena eseguito. Ecco alcune delle possibilità.

(a) un comando PRINT farà comparire qualche cosa sullo schermo

(b) un comando LET creerà, se necessario, una nuova variabile, attribuendole un valore. Sia PRINT sia LET sposteranno il puntatore di programma al successivo comando in sequenza cosicchè, quando il computer viene fatto ripartire, 'sa.' quale comando eseguire successivamente

(c) un GOTO non visualizzerà nulla nè altererà alcuna variabile, ma semplicemente ripristinerà il puntatore di programma in modo che indichi il comando citato nel GOTO. Per esempio il comando

130 GOTO 270

inserirà 270 nel contatore di programma

* *Se si togliesse il coperchio del computer non si vedrebbero effettivamente queste cose ma solo pochi chip di silicio e altri componenti. Comunque sia gli appropriati strumenti elettronici, mostrerebbero certamente le varie situazioni sopra citate.*

(d) il comando IF funzionerà allo stesso modo, salvo che viene elaborata per prima cosa la condizione. Se è *vera* il contatore di programma viene impostato esattamente come in un GOTO. Se è *falsa*, il contatore di programma viene semplicemente spostato sull'etichetta del successivo comando in sequenza

Osservare: 120 IF X=5 THEN 170
130 PRINT"NO"

Se X ha il valore 5, la condizione è vera e il contatore di programma viene modificato in 170. Per contro, se X ha qualche altro valore, il contatore di programma è semplicemente fatto avanzare a 130.

(e) il comando STOP indica che il programma è terminato visualizzando un messaggio BREAK. Non c'è scopo a continuare il programma oltre questo punto.

Per imitare accuratamente il computer, occorre poter vedere tutte queste parti chiaramente: il programma, le variabili, lo schermo video e il contatore di programma.

Un buon metodo consiste nell'usare una "tabella di tracciamento del programma" che è possibile disegnare su un pezzo di carta. La si potrebbe disegnare così:

CONTATORE DI PROGRAMMA 10

VARIABILI

SCHERMO	*PROGRAMMA*
	10 A=5
	20 PRINT "ALPHA="; A
	30 A=A*3
	40 B=A+37
	50 PRINT "BETA="; B
	60 STOP

Il programma che si intende "tracciare" viene scritto sulla destra e il valore iniziale del contatore di programma - e cioè il numero di etichetta del primo comando da eseguire - nella parte superiore. Assicurarsi che il programma sia una copia esatta di quello di cui si sta cercando il difetto: in caso contrario, il tracciamento rappresenta una perdita di tempo.

Ora, è possibile partire. Il contatore di programma contiene il valore "10", cosicche occorre prendere in considerazione e interpretare il comando contrassegnato "10", cioè A=5: si tratta di un comando LET. Cercare una A nella casella contrassegnata VARIABILI. Dopo aver riscontrato che questa non è ancora presente, scrivere una A, un due punti e il valore 5. Infine, spostare il contatore di programma sul successivo comando in sequenza, tracciando una riga sul valore precedente.

CONTATORE DI PROGRAMMA ~~1Ø~~ 2Ø

VARIABILI A: 5

SCHERMO	PROGRAMMA
	1Ø A=5
	2Ø PRINT "ALPHA="; A
	3Ø A=A★3
	4Ø B=B+37
	5Ø PRINT "BETA="; B
	6Ø STOP

Il successivo comando è interpretato allo stesso modo. Occorre dimenticare lo "scopo" del programma e tutte le informazioni riguardanti la sua sequenza e prendere in considerazione il comando 2Ø, soltanto perché ciò viene indicato dal contatore di programma. Il comando è un PRINT ed è possibile che esso visualizzi "ALPHA=5" Scrivere ciò nella parte SCHERMO e far avanzare il contatore di programma in modo da avere:

CONTATORE DI PROGRAMMA ~~1Ø~~ ~~2Ø~~ 3Ø

VARIABILI A: 5

SCHERMO	PROGRAMMA
ALPHA = 5	1Ø A=5
	2Ø PRINT "ALPHA="; A
	3Ø A=A★3
	4Ø B=A+37
	5Ø PRINT "BETA="; B
	6Ø STOP

Il successivo comando attribuisce un nuovo valore alla variabile A, che questa volta è già esistente. Occorre qui per prima cosa calcolare il valore dell'espressione A★3, usando il vecchio valore (5) della variabile A. Si prende poi nota del risultato, barrando il vecchio valore come qui indicato:

A: ~~5~~ 15

Il comando successivo crea una nuova variabile. Continuare il tracciamento fino a che non si raggiunge STOP. Il risultato finale è:

CONTATORE DI PROGRAMMA ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~5Ø~~ 6Ø

VARIABILI A: ~~5~~ 15 B: 52

SCHERMO	PROGRAMMA
ALPHA = 5	1Ø A=5
BETA = 52	2Ø PRINT "ALPHA="; A
BREAK IN 6Ø	3Ø A=A★3
READY.	4Ø B=A+37
	5Ø PRINT "BETA="; B
	6Ø STOP

Il successivo esempio comporta una semplice iterazione:

```
1Ø P=1
2Ø PRINT P;P★P★P
3Ø P=P+1
4Ø IF P<4 THEN 2Ø
5Ø STOP
```

Il tracciamento di questo programma fino alla riga 3Ø è lineare:

CONTATORE DI PROGRAMMA ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ 4Ø

VARIABILI P: ~~1~~ 2

SCHERMO	PROGRAMMA
1 1	1Ø P=1
	2Ø PRINT P; P★P★P
	3Ø P=P+1
	4Ø IF P<4 THEN 2Ø
	5Ø STOP

Il successivo comando in 4Ø è un IF. Per imitare il computer, valutare la condizione in cui 4 è maggiore di P. Dato che il valore corrente di P è 2 (come si può notare dalla sezione VARIABILI) e 2 è chiaramente minore di 4, la condizione è vera. Tutto ciò che occorre fare, pertanto, è inserire 2Ø come nuovo valore nel contatore di programma. Si ottiene:

CONTATORE DI PROGRAMMA ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ 2Ø	
VARIABILI P: ~~1~~ 2	
SCHERMO	*PROGRAMMA*
1 1	1Ø P=1
	2Ø PRINT P; P★P★P
	3Ø P=P+1
	4Ø IF P<4 THEN 2Ø
	5Ø STOP

Il tracciamento continua in questo modo fino a che la condizione risulta falsa, e il programma raggiunge STOP. Il risultato finale è:

CONTATORE DI PROGRAMMA ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ 5Ø	
VARIABILI P: ~~1~~ ~~2~~ ~~3~~ 4	
SCHERMO	*PROGRAMMA*
1 1	1Ø P=1
2 8	2Ø PRINT P; P★P★P
3 27	3Ø P=P+1
BREAK IN 5Ø	4Ø IF P<4 THEN 2Ø
READY.	5Ø STOP

Introdurre nel computer questo programma e lanciarlo. Verificare che i risultati corrispondano.

Ora introdurre di nuovo il programma per stampare ALPHA e BETA e lanciarlo.

Dovrebbe apparire:

ALPHA=5
BETA=52
BREAK IN 6Ø
READY

Il computer effettua molto facilmente i propri tracciamenti. Fare una prova battendo

TRON RETURN

(TRON sta per "TRACE ON") e lanciare di nuovo il programma. Ora i valori del contatore di programma appaiono tra parentesi quadre, in questo modo:

[1Ø] [2Ø] ALPHA =5
[3Ø] [4Ø] [5Ø] BETA =52
[6Ø]
BREAK IN 6Ø

Ciò informa che il programma esegue le righe 1Ø, 2Ø, 3Ø, 4Ø, 5Ø e 6Ø in quest'ordine.

L'output

ALPHA=5

è stato prodotto dalla riga 2Ø, come previsto. Per terminare il tracciamento battere

TROFF RETURN

Da questo momento in poi sarà possibile usare TRON e TROFF quale aiuto per le operazioni di tracciamento. Si noterà rapidamente che questi comandi vanno usati con prudenza, perché un programma contenente iterazioni generalmente esegue una gran quantità di comandi e la lista di numeri di riga può facilmente riempire lo schermo.

ESERCIZIO 8·1

Esercitarsi ora eseguendo il tracciamento dei seguenti programmi. Scrivere a matita e tenere a portata di mano una gomma per cancellare eventuali errori.

(a)

CONTATORE DI PROGRAMMA 1Ø

VARIABILI

SCHERMO	*PROGRAMMA*
	1Ø X=5
	2Ø Y=7
	3Ø Z=X+Y
	4Ø W=Y−X
	5Ø PRINT X;Y;Z;W
	6Ø STOP

(b)

CONTATORE DI PROGRAMMA 1Ø

VARIABILI

SCHERMO	*PROGRAMMA*
	1Ø Q=1
	2Ø PRINT "LEI MI AMA"
	3Ø PRINT "LEI NON MI AMA"
	4Ø Q=Q+1
	5Ø IF Q<3 THEN 3Ø
	6Ø STOP

Una volta completati questi due esercizi controllare le risposte eseguendo i programmi sul computer con la modalità di tracciamento attivata (TRON).

Esercizio 8.1 completato	

ESERCIZIO 8·2

Chiariamo ora come questo tipo di controllo venga utilizzato per trovare gli errori. Tutto dipende dall'alternarsi tra lo stato di obbedienza passiva tipico di un robot e lo stato di intelligenza umana. Innanzitutto occorre simulare il comportamento di un robot ed eseguire ogni comando esattamente come farebbe il computer. Quindi, tornando a considerare le cose dal punto di vista umano, ci si chiede "è ciò che mi aspettavo?". Se la risposta è affermativa si procede con il tracciamento, in caso contrario c'è una buona indicazione del perché il programma non funziona.

Ecco un semplice esempio. Supponiamo di aver scritto un programma per visualizzare la tabellina del 12. Lo schermo che ci si aspetta è

```
TABELLINA DEL DODICI
1★12=12
2★12=24
3★12=36
(fino a)
12★12=144
```

Il programma comprende tutte le parti necessarie: un'iterazione, un comando per visualizzare un titolo e un comando PRINT per visualizzare ciascuna riga della tabellina. Esso si presenterà nella forma:

```
10 PRINT"TABELLINA DEL DODICI"
20 P=1
30 P=P+1
40 IF P<13 THEN 30
50 PRINT P;"★12=";P★12
60 STOP
```

Quando si esegue questo programma, i risultati sono leggermente sconcertanti. Tutto ciò che si ottiene è

```
TABELLINA DEL DODICI
13★12=156
BREAK IN 60
READY
```

Che non certo quello che ci si aspettava! L'errore potrebbe essere perfettamente ovvio ma supponiamo di non riuscire ad individuarlo a prima vista. Occorre iniziare il tracciamento e dopo alcune fasi si ottiene quanto segue:

CONTATORE DI PROGRAMMA ~~10 20 30 40 30 40~~ 30

VARIABILI P: ~~1 2 3~~ 4

SCHERMO	PROGRAMMA
TABELLINA DEL DODICI	10 PRINT "TABELLINA DEL DODICI"
	20 P=1
	30 P=P+1
	40 IF P< 13 THEN 30
	50 PRINT P; "★12="; P★12
	60 STOP

A questo punto ci si rende conto che la variabile P viene incrementata fino a raggiungere il valore 12 senza che nulla venga visualizzato. È ora chiaro che il comando PRINT dovrebbe essere all'interno dell'iterazione e non all'esterno. Il posto giusto è tra i comandi 2Ø e 3Ø. Il comando IF a sua volta deve essere cambiato per saltare indietro al PRINT. Una piccola rapida correzione produce

```
10 PRINT "TABELLINA DEL DODICI"

20 P=1

25 PRINT P; "★12="; P★12

30 P=P+1

40 IF P < 13 THEN 25

60 STOP
```


Il tracciamento di un programma è una tecnica estremamente utile se si ha la pazienza di eseguirla passo per passo. Se si tira ad indovinare su intere sezioni di programma, si ripeterà probabilmente lo stesso errore commesso originariamente nello scrivere il programma, e il controllo non rivelerà alcun errore.

Per risparmiare tempo, è spesso possibile usare il meccanismo di tracciamento incorporato nei computer Commodore che indicherà la sequenza dei comandi eseguiti. Sarà tuttavia ancora necessario analizzare molto attentamente la lista dei numeri di riga - TRON non sostituisce il pensiero!

A volte si avrà la certezza che la maggior parte di un programma funzioni correttamente, quindi, per evitare di prendere in considerazione troppi numeri di riga - cioè anche quelli irrilevanti - basterà che la funzione di tracciamento sia attivata per una piccola parte del programma. Ciò si può ottenere numerando nel modo seguente i comandi TRON e TROFF nel programma stesso:

```
 10-
 20-
.........
 105 TRON  ⎫
 110-      ⎪
 120-      ⎪  Parte "tracciata"
.........  ⎬   del programma
 185 TROFF ⎭
 190-
 200-
```

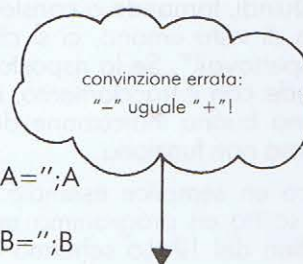
Una volta terminata la messa a punto degli errori della parte sospetta del programma, si potranno facilmente eliminare di nuovo i comandi TRON e TROFF.

Ci sono alcuni casi in cui il metodo di tracciamento sopra descritto non funziona ed è opportuno conoscerli:

- Se un programma è molto ampio, un controllo diretto richiederebbe troppo tempo. Metodi più appropriati saranno descritti in seguito, mano a mano che se ne presenterà la necessità. Se si parte con la convinzione di non aver commesso errori, l'operazione di tracciamento non sarà di molto aiuto.

Molte persone, dopo aver scritto l'ultima riga di un programma, credono infatti di averlo scritto tutto giusto. Tale sensazione, peraltro estremamente fuorviante, deriva semplicemente dalla mancata consapevolezza degli eventuali errori commessi. È molto più costruttivo pensare "probabilmente ho commesso degli errori, vediamo di trovarli". Ovviamente, questo richiede un piccolo sacrificio del proprio orgoglio!

- Se sono stati interpretati erroneamente alcuni aspetti fondamentali del BASIC, anche un controllo sarà di scarso aiuto. Per fare un esempio grossolano, si immagini qualcuno che creda fermamente – ma erroneamente – che nel BASIC il segno "–" significhi "addizione". Per addizionare due numeri questa persona scriverà il seguente programma:

```
10 A=34
20 B=19
30 PRINT "A=";A
40 PRINT "B=";B
50 PRINT "A PIÙ B=";A-B
60 STOP
```

Quando il programma viene eseguito, visualizza

```
A=34
B=19
A PIÙ B=15
BREAK IN 60
READY
```

che è chiaramente sbagliato. Per contro, quando lo si controlla, si trova che il comando 5Ø dà

```
A PIÙ B=53
```

che è quanto ci si aspetta. Finchè permane la convinzione che "–" significhi "sommare", non si troverà mai l'errore!

Naturalmente la maggior parte delle incomprensioni sono molto più sottili di questa. Ciononostante, se durante il controllo si ottengono ripetutamente risultati diversi da quelli visualizzati dal computer, è evidente che non si è compreso perfettamente qualcosa in materia di tecnica di programmazione. Se è possibile, farsi consigliare da qualcuno che conosca a fondo il BASIC*; altrimenti tornare all'inizio di questo manuale e controllare ogni argomento. Quasi sempre ciò porterà alla luce il problema.

Talvolta – sebbene molto raramente – la difficoltà può essere provocata da un guasto meccanico nel computer. Le apparecchiature moderne come il computer sono estremamente solide ed affidabili e quando si guastano è molto semplice evidenziare il problema: il cursore non compare quando si accende l'apparecchio o risulta impossibile caricare i programmi. In pratica non è quasi mai il caso di attribuire al computer la mancata esecuzione del programma fino a che non si è esaminata due o tre volte qualsiasi altra possibilità. Quando si invia la macchina per la riparazione, occorre spiegare esattamente perché si pensa che sia guasta ed allegare una copia del programma che non viene eseguito correttamente.

Ecco due programmi contenenti errori da trovare e da correggere.

(a) Supponiamo che questo programma visualizzi una tabella di conversione dei galloni in litri, iniziando con un gallone e terminando con 10 galloni (1 gallone=4.5 litri)

```
10 PRINT"GALLONI","LITRI"
20 G=1
30 PRINT G, 4.5★G
40 G=G+1
50 IF G>11 THEN 30
60 STOP
```

(b) Questo programma dovrebbe rappresentare una soluzione al problema 1 nella Lezione 7 per visualizzare un triangolo di stelle. È stato scritto da qualcuno che stava imparando il BASIC:

```
10 A$="★"
20 PRINT A$
30 A$="★★"
40 A$<>"★★★★★★★★★"THEN 20
50 STOP
```

Esercizio 8.2 completato	☐

* *Ci sono numerose persone che conoscono il BASIC. Non occorre conoscerle personalmente: basterà un'inserzione nel negozio locale per trovare rapidamente aiuto.*

Supponiamo che il programma UNIT8PROG debba visualizzare la tabellina del 7, ma contenga parecchi errori. Caricarlo, trovare e correggere gli errori. Controllare le risposte nell'Appendice B.

Esercizio 8.3 completato	☐

LEZIONE: 9

Verranno ora eseguiti altri disegni. Questa volta, si farà in modo che sia il computer ad eseguire il lavoro più impegnativo.

Nelle Lezioni 2 e 3, come probabilmente si ricorderà, si è parlato della possibilità di usare un certo numero di "funzioni" di controllo quando si eseguono disegni sullo schermo:

- Spostamento del cursore in quattro direzioni diverse
- Selezione di sedici colori diversi
- Inversione del colore e dello sfondo
- Caratteri lampeggianti
- Spostamento del cursore nella posizione "home" (angolo in alto a sinistra)
- Cancellazione del contenuto dello schermo

Più funzioni utilizzano a volte lo stesso tasto sulla tastiera, così che occorre usare i tasti SHIFT o CTRL o C= per selezionare la funzione richiesta.

Si ricorderà che è possibile impostare il colore della cornice e dello sfondo utilizzando istruzioni "COLOR" e i numeri di codice della tabella di pagina ??? .

Il computer è anche in grado di eseguire disegni sullo schermo sotto il controllo di un programma. Ogni programma può utilizzare tutte le funzioni di controllo schermo: può selezionare qualsiasi colore da assegnare ai caratteri, all'occorrenza può cancellare il contenuto dello schermo e può spostare il proprio cursore (che è invisibile all'utente) in qualsiasi posizione, utilizzando le funzioni di controllo cursore.

Naturalmente il calcolatore esegue queste operazioni solo in conseguenza di comandi precedentemente impartitigli.

Mettere sotto forma di comando le funzioni di controllo schermo è un'operazione semplice: devono essere incluse in stringhe insieme agli altri caratteri da visualizzare. In un primo momento ciò può sembrare strano. Infatti è comprensibile che ci si chieda se, battendo una stringa che includa una funzione di cancellazione schermo, l'intero schermo non scompaia durante la battitura. In effetti ciò non si verifica, come verrà dimostrato nell'esempio seguente.

ESERCIZIO 9.1

Nella Lezione 2 era stato momentaneamente sconsigliato l'uso delle virgolette. A questo punto ne verranno presi in esame gli effetti e risulterà quindi chiaro perché le virgolette sono così utili.

Quando si inizia a battere un comando (ad esempio dopo READY o RETURN) il computer si trova nella modalità "normale". Le funzioni di controllo come la selezione del colore o il movimento del cursore funzionano nel modo previsto. Non appena si batte un carattere virgolette (") per contrassegnare l'inizio di una stringa, il computer viene portato nella modalità virgolette. I caratteri normali, come le lettere o i simboli grafici vengono ancora trattati nel modo ordinario, ma non vengono più eseguite le funzioni di controllo: queste vengono disposte nella stringa come caratteri "speciali", rappresentati di solito su sfondo invertito. Il computer ritorna alla modalità normale quando si battono per la seconda volta le virgolette (per concludere la stringa) o se si batte RETURN .

Avviare il computer, battere un segno di virgolette e quindi far seguire ad una ad una tutte le funzioni di controllo.
(Osservare come ognuna di esse appare sullo schermo e compilare la tabella che segue)

Funzione	Tasto	Simbolo visualizzato
Cancellazione schermo	SHIFT e CLR HOME	
Cursore in posizione di partenza	CLR HOME	
Cursore verso l'alto	↑	
Cursore verso il basso	↓	
Cursore a sinistra	←	
Cursore a destra	→	
Nero	CTRL e 1 BLK	
Bianco	CTRL e 2 WHT	
Rosso	CTRL e 3 RED	
Viola	CTRL e 4 CYN	
Porpora	CTRL e 5 PUR	
Verde	CTRL e 6 GRN	
Blu	CTRL e 7 BLU	
Giallo	CTRL e 8 YEL	
Arancio	C= e 1 ORNG	
Marrone	C= e 2 BRN	
Giallo-Verde	C= e 3 YL GRN	
Rosa	C= e 4 PINK	
Verde-Grigio	C= e 5 BL GRN	
Blu chiaro	C= e 6 L BLU	
Blu scuro	C= e 7 D BLU	
Verde chiaro	C= e 8 L GRN	
Reverse on	CTRL e RVS ON	
Reverse off	CTRL e RVS OFF	
Flash on	CTRL e Flash on	
Flash off	CTRL e Flash off	

Provare ora a vedere il funzionamento di alcuni di questi comandi. Per prima cosa assicurarsi che l'apparecchio televisivo sia regolato correttamente per ricevere il colore usando, se necessario, il programma TESTCARD. Assicurarsi che lo sfondo sia bianco battendo il comando COLORØ,2 RETURN.

Quindi visualizzare sullo schermo la parola "EDINBURGH" in blu. Battere il comando

Ciò che appare effettivamente sullo schermo (ancora tutto in nero) è

Il simbolo ← invertito è il codice del colore blu.

Premere ora il tasto RETURN. Sullo schermo appare la parola EDINBURGH, in blu.

Questo Esercizio illustra abbastanza chiaramente il principio che quando si batte all'interno di una stringa una funzione di controllo, quest'ultima non viene attivata nel momento in cui è battuta, ma solo quando la stringa viene visualizzata dal computer.

Si noterà che il cursore lampeggiante è rimasto blu. Cambiare in nero il colore battendo, senza virgolette, la corretta funzione di controllo.

Un comando PRINT contenente un cambiamento di colore può essere inserito in un programma, proprio come qualsiasi altro comando.

Introdurre ed eseguire quanto segue:

4Ø STOP

Il comando 3Ø dimostra la possibilità di inserire in una stringa più di una funzione di controllo; ed anche che le lettere gialle si distinguono a fatica su uno sfondo bianco.

Anche le funzioni di controllo schermo e di controllo cursore possono essere disposte in stringhe. Battere quanto segue:

→ → CTRL e 3 RED PARIS"

Sullo schermo tutto questo apparirà nella forma seguente

Quando si batte RETURN, le funzioni di controllo vengono effettivamente eseguite. Il contenuto dello schermo viene cancellato, il cursore si sposta di tre posizioni verso il basso e di tre posizioni verso destra e circa a metà schermo compare in rosso la parola PARIS. Verificare questo applicando l'esempio fornito.

In generale, è possibile fare in modo che il computer visualizzi parole e simboli dovunque si desideri includendo in una stringa il numero appropriato di movimenti del cursore. Quando si fa in modo che il computer esegua un disegno sullo schermo, naturalmente non si desidera che il lavoro compiuto venga rovinato quando viene visualizzato il messaggio READY ed il cursore lampeggiante. Un modo per sormontare questo inconveniente è di utilizzare un "arresto a circolo chiuso", o una funzione GOTO che ritorni sempre su se stessa. Una volta che il computer raggiunge questo comando, entrerà in "loop" e non visualizzerà READY finché non verrà premuto il tasto RUN STOP. Questo programma, ad esempio, visualizzerà la parola LONDON in bianco al centro di uno schermo nero:

1Ø COLORØ,1

3Ø GOTO 3Ø

Introdurre questo programma, avviarne l'esecuzione e quindi arrestarlo con il tasto RUN STOP. Lo schermo rimarrà nero ed il cursore bianco, ma è possibile ripristinare la condizione normale tenendo premuto RUN STOP e premendo il tasto RESET posto sulla parte laterale del computer.

Quando appare la parola MONITOR, battere X e RETURN. Si ricorderà che è sempre

possibile eseguire questa operazione se il computer per qualsiasi motivo si arresta; è preferibile utilizzare questo sistema piuttosto che accendere e spegnere il computer, operazione che provocherebbe la perdita del programma.

Per esercitarsi, fare in modo che il computer visualizzi parole e disegni di diversi colori in varie posizioni dello schermo. Ricordare che la funzione SHIFT e CLR HOME cancella il contenuto dello schermo, perciò se il programma contiene una sequenza di istruzioni PRINT, solo la prima dovrebbe iniziare con questa funzione - mentre alcune altre possono cominciare con la funzione .

Esercizio 9.1 completato

È noto che i moderni orologi sono controllati da cristalli al quarzo e mantengono una notevole precisione anche per lunghi periodi di tempo. Anche il computer incorpora un cristallo al quarzo che vibra a diversi milioni di volte al secondo che viene utilizzato – fra le altre cose – per controllare un orologio interno digitale. Questo orologio non possiede un proprio quadrante, ma viene trattato esattamente come una variabile stringa così che è possibile visualizzare l'ora sullo schermo quando si desidera. Il nome della variabile dell'orologio è TI$.

Quando si avvia per la prima volta, qualsiasi orologio va impostato sull'ora esatta. Il **16** e il **PLUS/4** non fanno eccezione. L'orologio si può regolare tramite la tastiera, battendo un comando del tipo

TI$="193746" RETURN

In questo caso l'orologio viene impostato sulle 7.37 e 46 secondi di sera.

Se si vuole impostare l'orologio con estrema precisione, regolarsi con il segnale orario dato dalla radio o dalla televisione, ad esempio quello delle nove. Poco prima che venga data l'ora esatta battere

TI$="Ø9ØØØØ"

e quindi battere RETURN non appena si sente l'ultimo segnale acustico.

Una volta che l'orologio interno del computer è stato regolato, segnerà il tempo con uno scarto di alcuni secondi al giorno, finchè l'apparecchiatura non venga spenta o reinizializzata. Non occorre ripristinarlo o aggiornare l'ora dall'interno di un programma.

Per visualizzare l'orario, è sufficiente menzionare la variabile TI$ in un comando PRINT.

Ora regolare l'orologio del computer, usando il proprio orologio (non ha importanza se la regolazione non è molto precisa). Quindi visualizzare più volte il valore di TI$, utilizzando un comando PRINT. Si osservi come l'orologio tiene conto del trascorrere dei secondi anche quando non è presente sullo schermo.

Ora mantenere sullo schermo la visualizzazione dell'ora, eseguendo il programma

```
10 PRINT TI$
20 GOTO 10
```

Arrestare questo programma, attendere alcuni minuti e riavviarlo. Si potrà notare che l'ora è ancora esatta e che l'orologio ha funzionato ininterrottamente per tutto il tempo.

Questo metodo di visualizzazione dell'ora non è particolarmente efficace. È invece possibile fare in modo che il computer fornisca le prestazioni di un autentico orologio digitale tramite il seguente programma:

comando 10: Seleziona una cornice porpora
comando 20: Seleziona uno sfondo giallo
comando 30: Cancella il contenuto dello schermo
comando 40: Sposta il cursore nella posizione "home", quindi verso il basso di 9 righe e verso destra di 6 posizioni; non occorrono nuove righe
comando 50: Visualizza TI$ in blu
comando 60: Salta indietro al comando 40.

Scrivere nel riquadro qui di seguito il codice di questo programma; quindi batterlo sulla tastiera del computer e provarlo. Se si incontrano serie difficoltà, controllare nella Appendice B la versione corretta, e studiarla attentamente fino alla totale comprensione del suo funzionamento.

Esercizio 9.2 completato

ESERCIZIO 9·3

L'uso dei cicli (loop) controllati è spesso utile quando si disegnano figure sullo schermo. Supponiamo che si desideri disegnare un blocchetto di 10x10 punti rossi nell'angolo in alto a sinistra. Ciò può essere eseguito visualizzando dieci righe, ognuna con dieci simboli grafici ●:

```
10 PRINT" SHIFT e CLR HOME ";

20 J=1

30 PRINT" CTRL e 3 RED ●●●●●●●●●●"

40 J=J+1

50 IF J<11 THEN 30

60 GOTO 60
```

Questo programma riassume alcuni dei concetti già espressi nelle Lezioni precedenti. Il punto e virgola al termine del comando 10 impedisce al computer di iniziare una nuova riga dopo aver cancellato lo schermo, così che la prima riga di punti rossi appare all'estremità superiore dello schermo stesso.

Le istruzioni dalla 20 alla 50 formano un "loop" controllato e l'istruzione 60 costituisce un arresto ciclico in quanto continua a richiamare se stessa.

Introdurre il programma ed eseguirlo così come mostrato nell'esempio. Quindi arrestarlo e verificare di persona gli effetti delle seguenti operazioni:

a) eliminazione del punto e virgola dopo "CLR HOME"
b) sostituzione con un valore diverso (ad esempio 15) del valore 11 contenuto nel comando 50
c) eliminazione del comando 60

Naturalmente questi cambiamenti possono essere eseguiti con le funzioni di lista (LIST), cancellazione e modifica. Ricordare di riportare di nuovo a blu o nero il colore del cursore prima di iniziare.

Per ottenere un blocchetto pieno di colore omogeneo, si usano gli spazi invertiti. Provare a cambiare la riga 3Ø in

PRINT " CTRL e 3 RED CTRL e RVS ON ← 10 spazi →"

ed eseguire di nuovo il programma.

Può essere interessante osservare cosa accade se si desidera più di un blocchetto colorato nello stesso disegno. La procedura da adottare è di spostare il cursore alla prima riga dell'area e quindi riempirla, senza interferire con il colore già presente sullo schermo.

Prendere in esame i due esempi seguenti:

(a) Per visualizzare un blocchetto blu 1Øx1Ø al di sotto del blocchetto rosso:

La parte inferiore dello schermo è vuota, non c'è quindi alcun rischio di danneggiare un lavoro precedentemente eseguito. Inoltre, dopo aver eseguito il disegno del blocchetto rosso, il cursore si troverà nella posizione appropriata. È possibile ampliare il programma con l'aggiunta di:

6Ø J=1

7Ø PRINT " CTRL e 7 BLU CTRL e RVS ON ← 10 spazi →"

8Ø J=J + 1

9Ø IF J< 11 THEN 7Ø

1ØØ GOTO 1ØØ

Notare che l'arresto ciclico (1ØØ GOTO 1ØØ) è stato spostato alla fine del programma. La variabile di controllo J viene usata sia nel "loop" del rosso sia in quello del blu: ciò è corretto, dal momento che l'esecuzione del blocchetto rosso è terminata, prima che inizi quella del blocchetto blu e a J non vengono attribuite due funzioni contemporaneamente.

(b) Per visualizzare un blocchetto nero 1Øx1Ø accanto al blocchetto rosso:
La riga iniziale è quella che si trova nella parte superiore, quindi nell'esecuzione del disegno dell'area nera è necessario prestare attenzione a non danneggiare il blocchetto rosso, che si trova già sullo schermo. Ciò può essere ottenuto spostando il cursore nella posizione "home" e visualizzando 1Ø righe, ognuna della quali cominci con 1Ø spostamenti del cursore verso destra, per oltrepassare il blocchetto rosso.
L'ampliamento del programma avviene nel modo seguente:

A questo punto unire assieme le varie parti del programma, batterlo e provarlo. Notare che è costituito da tre "loop" distinti che vengono eseguiti uno di seguito all'altro.

Provare ad ampliare il programma, in modo da disporre un blocchetto porpora sotto il blocchetto nero...

Come Esercizio finale, provare a scrivere programmi per la visualizzazione di bandiere dal disegno non troppo articolato, o altre figure che riempiano lo schermo completamente. Prestare la massima attenzione nella realizzazione di questi programmi, in quanto è possibile imbattersi in numerose difficoltà inaspettate.

- Generalmente il significato di un punto e virgola al termine di un comando PRINT è di impedire che venga cominciata una nuova riga. Se si imposta il computer in modo che visualizzi un carattere nella posizione di estrema destra di una riga, automaticamente il cursore verrà spostato nella riga successiva. Pertanto le visualizzazioni destinate a riempire un'intera riga devono essere seguite da punti e virgole, a meno che non si desideri che la riga seguente sia vuota.
- Non è possibile usare un comando PRINT per visualizzare un carattere nell'angolo in basso a destra dello schermo senza far spostare l'intero schermo verso l'alto.
Il modo in cui si può assegnare a questo quadrato il colore appropriato è di selezionare conformemente il colore dello sfondo.

È consigliabie pianificare accuratamente i disegni che si intendono visualizzare, con l'aiuto di un foglio di carta a quadretti. Quando si scrivono per le prime volte dei programmi, è inevitabile commettere numerosi errori e dover compiere diversi tentativi prima di giungere alla versione corretta.

In questi tentativi ripetuti risiede tuttavia l'essenza stessa dell'apprendimento.
Pertanto sarebbe controproducente desistere di fronte alle prime difficoltà incontrate.

Per cominciare, verrà fornito l'esempio di un programma per la realizzazione del disegno della bandiera francese.

blu	bianco	rosso

Assegnare alla striscia bianca centrale una larghezza di 14 caratteri, e alle altre due una larghezza di 13 caratteri. 13+14+13=4Ø

All'inizio, i colori appropriati per lo sfondo e la cornice possono essere, ad esempio, rispettivamente il rosso e il nero.

È possibile eseguire il disegno della bandiera visualizzando 25 righe, ognuna con tredici quadrati bianchi e quattordici blu. Ricordare che l'ultima riga deve essere trattata a parte poiché non deve essere seguita da una nuova riga. È possibile disporre le prime 24 righe in un "loop" controllato, ma l'ultima riga richiederà un comando apposito.

Si giunge quindi ad avere

```
1Ø COLOR4,1
2Ø COLORØ,3
```

```
3Ø PRINT" [SHIFT] e [CLR HOME] ";
4Ø J=1
5Ø PRINT" [CTRL] e [RVS ON] [CTRL] e
[7 BLU] ← 13 spazi → [CTRL] e
[2 WHT] ← 14 spazi →"
6Ø J=J+1
7Ø IF J<25 THEN 5Ø
8Ø PRINT" [CTRL] e [RVS ON] [CTRL] e
[7 BLU] ← 13 spazi → [CTRL] e
[2 WHT] ← 14 spazi →";
9Ø GOTO 9Ø
```

Eseguire questo programma e studiarlo attentamente fino a che non si comprenda il significato di ogni simbolo. Provare ora a realizzare il disegno di alcune bandiere evitando, per il momento, quelle con elementi diagonali. Provare la bandiera dell'Islanda, che appare a pagina 23 . È possibile controllare la propria risposta con quella fornita nell'Appendice B.

Esercizio 9.3 completato	

Il nome del programma di autotest per questa Lezione è UNIT9QUIZ.

LEZIONE: 10

Nelle precedenti Lezioni è stato chiarito che i comandi possono essere scritti una volta sola ma eseguiti ripetutamente parecchie volte. Ciò si verifica ogni volta che in un comando si inserisce una iterazione.

Su scala molto più ampia, una cosa simile avviene con i programmi completi. La maggior parte dei programmi è studiata per essere utile, il che significa che essi sono memorizzati e distribuiti su nastri o su dischi o su ROM e usati molte volte da persone diverse. A titolo di esempio, osservare i vari programmi registrati su nastro che formano parte di questo corso.

Iniziamo considerando tutti i programmi che sono stati finora scritti personalmente. Ognuno di essi presenta l'inconveniente di produrre sempre lo stesso risultato ogni volta che viene eseguito. Difficilmente un programma del genere risulterebbe molto utile!

Per fare un esempio specifico, torniamo indietro al programma che calcola e visualizza un tabella di conversione tra le lire sterline e le lire italiane. Il programma era così strutturato:

```
10 PRINT"£","LIRE"
20 PRINT
30 PS=5
40 PRINT PS,2350★PS
50 PS=PS+5
60 IF PS<80 THEN 40
70 STOP
```

Il giorno in cui è stata scritta la Lezione interessata, il tasso di cambio era realmente 2350 lire per sterlina cosicchè il programma avrebbe dato risultati corretti. Il tasso di cambio è poi sceso a 2175. Qualsiasi banca che usasse questo programma originale per vendere lire in cambio di sterline, accumulerebbe delle grosse perdite.

Come ovviare a questa difficoltà? La soluzione più immediata sarebbe quella di variare la riga 40 in modo da farla apparire

```
40 PRINT PS,2175★PS
```

Sfortunatamente questa idea non porterebbe molto lontano. La maggior parte di coloro che usano il computer non sono programmatori o per lo meno non arrivano ad affrontare problemi così sofisticati.

Per rendere i programmi più flessibili, più adattabili alle esigenze quotidiane, occorre una nuova funzione: una funzione che consenta all'utente di fornire informazioni che il programmatore non poteva conoscere nel momento in cui il programma era stato scritto.

Un programma contenente una funzione di questo tipo può servire a perecchie persone e consentire a ciascuna di esse di risolvere una propria particolare versione di un problema. Per esempio, supponiamo che il programma di conversione delle monete consenta all'utente di indicare il tasso corrente di cambio ogni volta che viene usato; in tal caso diventerebbe immediatamente utile a tutte le banche del mondo e funzionerebbe correttamente per qualsiasi possibile tasso di cambio.

Supponiamo ancora di progettare un programma che debba essere usato da altri. Si inizia decidendo quali quantità occorre lasciare indefinite, facendo in modo che il programma stesso chieda all'utente di fornire i valori relativi.

Nell'esempio in questione, il tasso di cambio è chiaramente una quantità di questo tipo: esso può essere sconosciuto al programmatore, ma non all'utente.

Si assegnano le quantità sconosciute a variabili a cui vengono attribuiti nomi adatti: ad esempio, per il tasso di cambio potrebbe essere RE (Rate Exchange). È possibile quindi scrivere un programma usando nomi simbolici invece dei valori effettivi (che non possono essere noti in anticipo). Pertanto la riga 40 del programma di cambio potrebbe comparire come segue:

```
40 PRINT PS,RE★PS
```

Naturalmente manca qualcosa a questa descrizione. Non si possono conoscere i valori delle variabili, ma la macchina deve conoscerli quando esegue il programma. Il comando che consente all'utente di inserire le informazioni mancanti ha la parola chiave INPUT. Questa è seguita dal nome (o dai nomi) delle variabili necessarie. Quando il comando INPUT viene eseguito, esso attende che l'utente batta un valore, che memorizza quindi nella variabile specificata. Ora può essere eseguito il resto del programma che usa questa variabile.
Prima di dare un esempio, è bene sottolineare un punto estremamente importante: ogni programma contenente un comando INPUT deve comunicare esattamente all'utente le informazioni richieste. Ciò può essere solitamente fatto con le istruzioni PRINT.

ESERCIZIO 10·1

Studiare accuratamente il seguente programma:

```
3 PRINT " BATTI IL TASSO DI "
4 PRINT "CAMBIO ODIERNO"
5 PRINT "TRA £ E LIRE"
6 INPUT RE
10 PRINT "£","LIRE"
20 PRINT
30 PS=5
40 PRINT PS, RE★PS
50 PS=PS+5
60 IF PS<80 THEN 40
70 STOP
```

Si noterà che il programma non attribuisce alcun valore particolare al tasso di cambio, ma usa la variabile RE per rappresentare questo valore ogni volta che se ne presenta la necessità. Il programma inizia richiedendo all'utente di fornire il valore necessario, facendo precedere questa richiesta da una breve spiegazione. Immettere il programma, controllarlo attentamente e battere RUN. Immaginiamo ora di essere un utente del programma: ad esempio, un cambiavalute che non sa nulla di programmazione. Tramite lo schermo la macchina chiede di battere qualche cosa, per cui occorre inserire la cifra appropriata e quindi battere

RETURN

Non appena si è eseguita questa operazione, lo schermo si riempie con una tabella di conversione che consente di iniziare a lavorare.

Eseguire il programma parecchie volte e notare come può gestire diversi tassi di cambio. Anche se la lira venisse rivalutata ad un livello di 23.7, rispetto alla sterlina, il programma produrrebbe sempre risposte esatte.

A questo punto si può nuovamente considerare il problema dal punto di vista del programmatore. Mentre il programma era in funzione, la visualizzazione del cursore e l'attesa dell'inserimento di informazioni da parte dell'utente rappresentava in effetti l'esecuzione del comando INPUT.

Il comando INPUT compare in parecchie forme leggermente diverse. Verrà ora dato qualche esempio e saranno citate alcune regole generali.

1. Cancellare la memoria battendo NEW e battere quindi

```
10PRINT"DIMMI IL TUO NOME"
20INPUT, N$
30PRINT"CIAO [SPACE] ";N$;"!"
```

Eseguire questo programma e osservare il risultato. L'esempio mostra come funziona il comando INPUT con stringhe e numeri. È possibile usare questa sequenza – o una analoga – nella parte iniziale di qualsiasi programma se si vuole che il computer si comporti in modo 'confidenziale' nei confronti dell'utente.

Nel caso di un programma di quiz, si sarebbe potuto usare un comando del tipo

```
40PRINT"NO [SPACE] ";N$;"PUOI FARE
    DI MEGLIO"
```

(Se rimangono dei dubbi su ciò che viene visualizzato da questo comando basta aggiornarlo alla fine del programma che si trova in memoria ed eseguire il programma di nuovo).

2. Provare

```
10INPUT"NOME";N$
20PRINT"CIAO [SPACE] "; N$
```

Questo esempio mostra come è possibile includere nel comando INPUT una breve spiegazione che compare sullo schermo come guida per l'utente, immediatamente prima del punto di domanda.

Il comando 10 nell'esempio è equivalente alla sequenza

```
PRINT"NOME";
INPUT N$
```

Notare che la stringa di parole descritte deve essere seguita da un punto e virgola.

3. Provare infine

```
10 PRINT "INSERIRE DUE NUMERI DA
    SOMMARE"
20 INPUT A,B
30 PRINT "SOMMA=";A+B
```

```
40 STOP
```

Il comando INPUT si aspetta ora due valori e l'utente deve batterli separati da una virgola o da un RETURN (Cioè potrebbe battere ad esempio 43,19

oppure 43
19)

In generale, il comando INPUT può chiedere all'utente qualsiasi numero di variabili. È comunque consigliabile che le variabili richieste non siano più di due per evitare confusione. Nell'ambito del comando, i numeri delle variabili sono separati da virgole. Dopo aver eseguito questo programma alcune volte, simulare il comportamento di un utente poco intelligente, provando a battere qualcosa di non significativo, ad esempio

PAPERINO, TOPOLINO

Il computer accetta come stringa qualsiasi sequenza di caratteri, ma se sta richiedendo l'introduzione di un numero e gli viene dato qualcosa che non può essere un numero, visualizza il messaggio

REDO FROM START

fornendo la possibilità di effettuare un altro tentativo. Se si vuole interrompere un programma mentre sta eseguendo una comando INPUT e visualizzando il cursore, dal momento che il tasto RUN STOP è disabilitato e quindi l'interruzione del programma è più complessa del solito, tenere premuto il tasto RUN STOP premendo il pulsante RESET. Il cursore apparirà sullo schermo dove sono visualizzate solo la parola MONITOR e alcune lettere e cifre.

Ora battere X RETURN ; apparirà il messaggio READY., a dimostrazione che tutto è di nuovo sotto controllo. Attenzione a non premere RESET da solo, altrimenti il programma verrà distrutto.

Esercizio 10.1 completato

ESERCIZIO 10·2

È abbastanza semplice scrivere programmi utili, purché si ricordi che il programmatore e l'utente sono due persone diverse. Non si può dare per scontato che l'utente sia un esperto di programmazione (quindi non ci si può aspettare che usi un comando LIST per scoprire il funzionamento del programma).

In generale, il programmatore non può comunicare con l'utente, se non facendo in modo che il computer visualizzi dei messaggi sullo schermo; inoltre, l'utente non può assolutamente comunicare con il programmatore, per cui è bene che il computer non lasci questioni in sospeso.

Paradossalmente, chi scrive un programma, potrebbe immaginare di essere una mosca poggiata sul muro di una stanza in cui qualcuno sta cercando di usare il programma stesso. A questo punto verrebbe spontaneo tentare di prevedere ogni possibile difficoltà e predisporre gli strumenti necessari per superarla, facendo in modo che l'utente sia guidato il più possibile.

Una volta che il programma è stato scritto, è possibile provarlo mettendosi al posto dell'utente; successivamente, come prova finale, sarebbe opportuno chiedere ad un amico o ad un parente di fare da cavia, facendo provare anche a lui un programma. Se la cavia deve porre domande su cosa fare o sul significato delle risposte visualizzate, significa che la prova non è riuscita e che occorre di conseguenza riprogettare il programma.

Scrivere ora due programmi che svolgano le seguenti operazioni:

(a) Visualizzare qualsiasi tabellina di moltiplicazione a scelta dell'utente.

(b) Chiedere all'utente (che si presume sposato) di specificare il proprio cognome e il nome di battesimo della moglie; dopodiché il programma deve visualizzare il nome completo della moglie.

Le soluzioni sono indicate nell'Appendice B, ma non consultarle fino a che non si è fatto tutto il possibile per scrivere questi programmi da soli.

Esercizio 10.2 completato	

Il quiz per questa Lezione è UNIT10QUIZ.

LEZIONE: 11

Una delle caratteristiche più interessanti della programmazione è costituita dalla vastissima gamma di applicazioni possibili. Lo stesso computer, se opportunamente programmato, può servire da calcolatrice, da mezzo didattico, da strumento musicale, da monitor per controllare le condizioni di un paziente in un ospedale e può essere utilizzato per un gran numero di altri scopi. Questa versatilità deriva dall'enorme numero di modi in cui è possibile combinare pochi comandi fondamentali.

Finora il vocabolario totale dei comandi che sono stati usati all'interno dei programmi è costituito da sette elementi:

PRINT, LET, GOTO, IF, INPUT, STOP e COLOR

Naturalmente ci sono altri comandi BASIC che occorrerà imparare, ma in questa Lezione ci si limiterà ad esplorare il potenziale dei comandi già noti. Il comando più flessibile di tutti è IF. Nelle precedenti Lezioni è stato usato per controllare le iterazioni ma è anche utile in molti altri casi. Per esempio, può controllare dati o elementi di informazione forniti dall'utente in modo da far compiere al computer il tipo di azione appropriata.

Immaginare di voler organizzare un'agenzia matrimoniale computerizzata. Per prima cosa occorrerà approntare un programma per fornire ai clienti consigli sull'età del partner adatto. Per tradizione, un uomo dovrebbe sposare una ragazza con un'età pari alla metà della sua più sette. E ciò implica, ovviamente, che una ragazza deve cercare un marito con il doppio della sua età meno 14.

Chiaramente, il programma deve iniziare chiedendo l'età del cliente. Quindi, per dargli il consiglio giusto, deve scoprire se il cliente è un uomo o una donna. Il programma verrà usato da uomini e donne, quindi deve comprendere un gruppo di comandi separati per fornire consigli a ciascuno dei due sessi. Inoltre, deve essere previsto un comando IF per scegliere il gruppo di comandi adatto al sesso del cliente. Qui di seguito è indicata la prima versione del programma. Studiarla accuratamente e spiegare in dettaglio il significato di ciascun comando:

```
10 INPUT "QUANTI ANNI HAI";AG
20 INPUT "MASCHIO O FEMMINA";SX$
30 IF SX$="MASCHIO" THEN 70
40 PRINT "DEVI CERCARE"
50 PRINT "UN UOMO DI";2★AG–14;ANNI
60 STOP
70 PRINT "DEVI TROVARE"
80 PRINT "UNA RAGAZZA DI";AG/2+7;
   "ANNI"
90 STOP
```

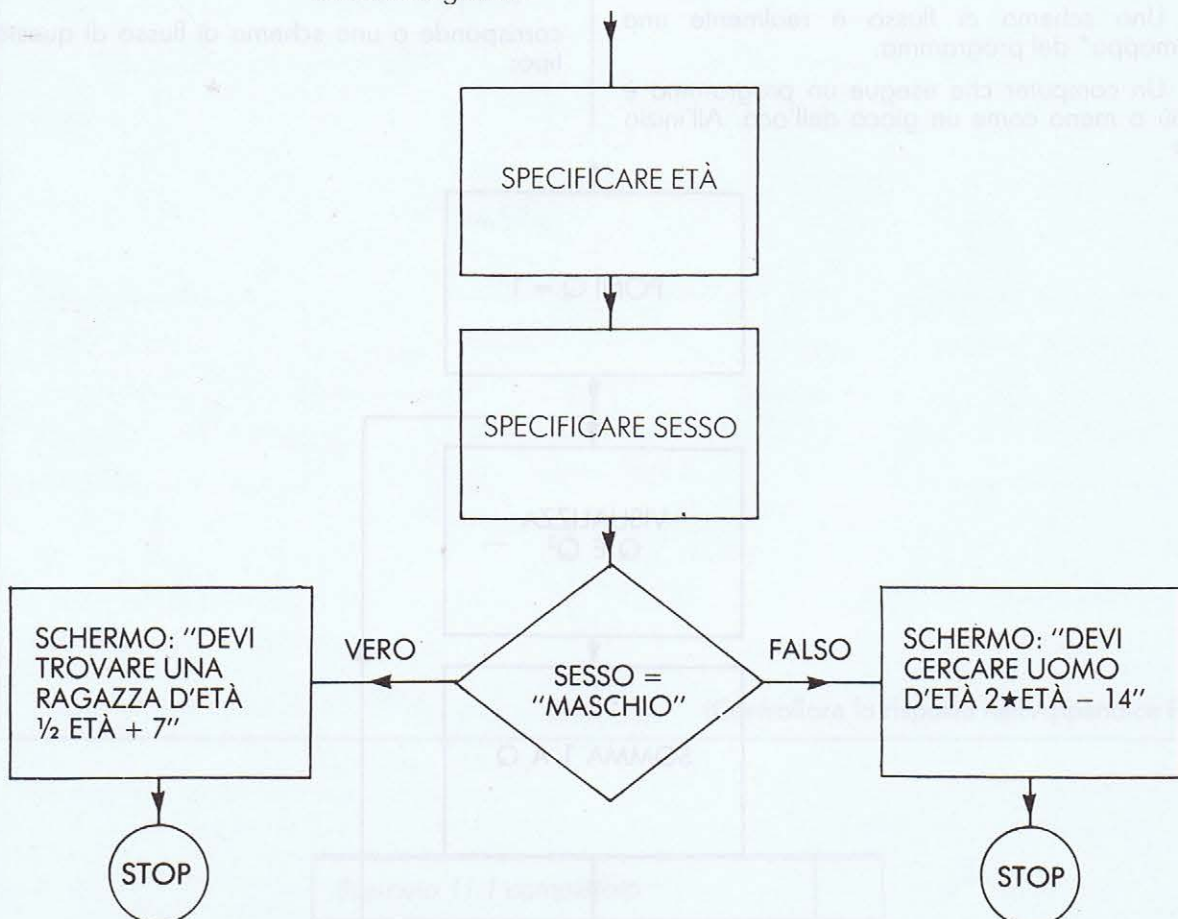
Si sarà notato l'uso della variabile AG per contenere l'età del cliente SX$ per memorizzare se è maschio o femmina. La condizione SX$="MASCHIO" è vera se il cliente risponde MASCHIO alla domanda "MASCHIO O FEMMINA?". L'espressione AG/2+7 è il modo con cui il basic esprime "metà dell'età + 7" e 2★AG–14 significa "due volte l'età meno 14". Una volta osservato il programma, provare a *prevedere* il più accuratamente possibile cosa comparirà sullo schermo:

(a) per un uomo di 20 anni e (b) per una ragazza di 22. Usare i riquadri che seguono. Il primo riquadro è parzialmente compilato.

(a)

(b)

Immettere il programma nel computer. Provarlo usando vari tipi di clienti e controllare che entrambe le previsioni siano corrette.

Questo semplice esempio dimostra che l'azione del computer non deve essere fissata in anticipo dal programmatore ma può essere resa dipendente dalle informazioni fornite dall'utente. I programmi devono spesso prendere complicate serie di decisioni cosicchè per pianificarli è opportuno usare un tipo speciale di diagramma detto schema di flusso. Lo schema di flusso per un programma di questo genere è il seguente:

Un diagramma di flusso si compone di un certo numero di blocchi collegati da righe con frecce. Ci sono quattro tipi di blocchi:

(a) Un quadrato o un rettangolo contiene la descrizione di un'azione semplice che può essere successivamente tradotta in uno o due comandi BASIC. Nello schema di flusso esemplificativo i due rettangoli superiori sono esempio di questo tipo. Le righe con frecce mostrano che il programma inizia eseguendo il primo blocco quindi passa al secondo nell'ordine.

(b) Un rombo contiene una condizione che può essere vera o falsa. Nel rombo entra una linea ma ne escono sempre due, contrassegnate VERO e FALSO (o talvolta SI e NO). Il rombo corrisponde ad un comando IF ed ordina al computer di verificare la condizione e di seguire la linea VERO o FALSO a seconda del risultato.

(c) La figura terminale, che ordina al computer di interrompere l'esecuzione del programma è un piccolo cerchio con la parola STOP.

(d) La nuvoletta (che non compare nell'esempio), è il simbolo di un'azione troppo complicata per poterla scrivere in dettaglio. Solitamente la nuvoletta potrebbe essere ampliata in un altro schema di flusso completo, esattamente come si farebbe con una mappa stradale nazionale che viene affiancata da mappe dettagliate delle diverse città.

Uno schema di flusso è realmente una "mappa" del programma.

Un computer che esegue un programma è più o meno come un gioco dell'oca. All'inizio la pedina del giocatore (un'automobile, un cappello a cilindro o qualsiasi altra cosa) va sul primo blocco. Ogniqualvolta l'azione descritta in un blocco è stata completata, la pedina si sposta, lungo la freccia, al blocco successivo.

Quando la pedina incontra un rombo, il giocatore esamina la condizione e decide se è vera. Se lo è, sposta la sua pedina al termine della linea VERO altrimenti segue la linea FALSO. Ad un certo punto raggiunge un blocco STOP che rappresenta la fine della partita.

Lo scopo di questo esempio figurato è quello di mettere in evidenza due cose che è importante comprendere riguardo ai computer:

- Un computer può eseguire soltanto una (ed una sola) azione alla volta.
- L'ordine in cui il computer esegue le azioni è determinato dal programma.

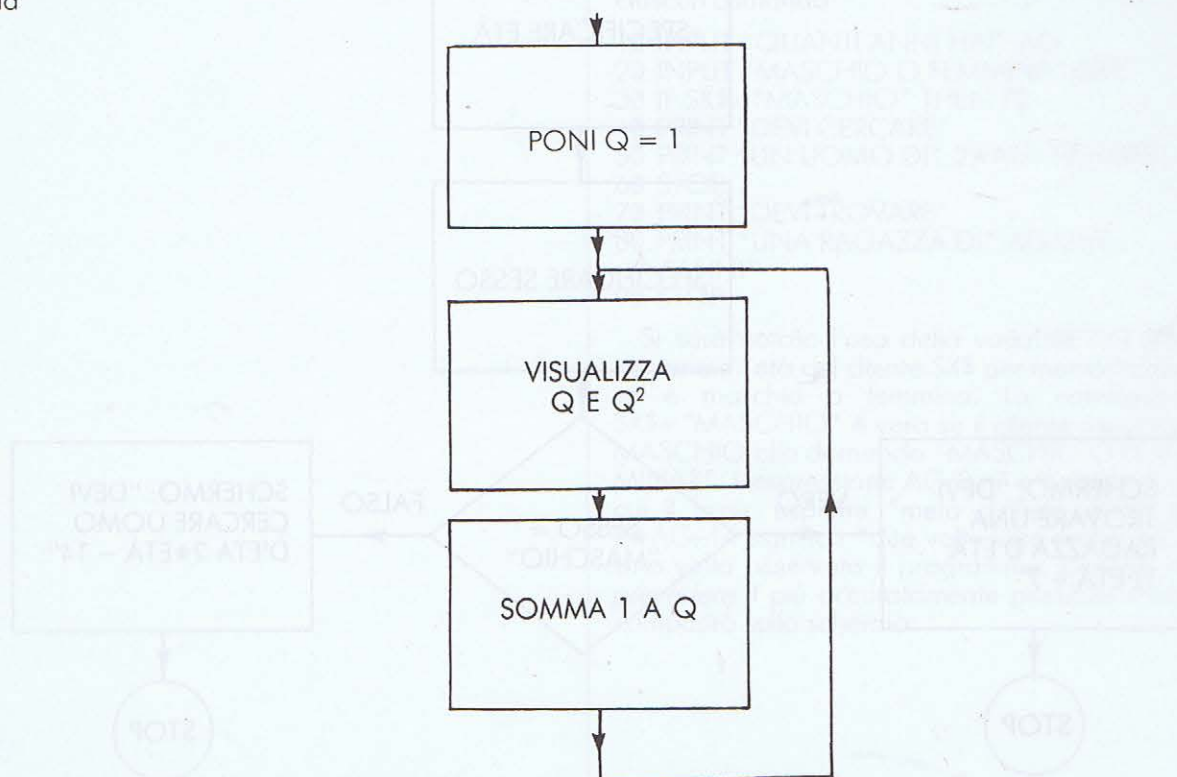
Ci si potrebbe chiedere perché negli schemi di flusso non è previsto un simbolo per il comando GOTO.

Ciò avviene perché GOTO non specifica alcuna azione, ma riguarda soltanto l'ordine in cui i comandi vengono eseguiti.

La sua rappresentazione è quindi una semplice linea di collegamento.

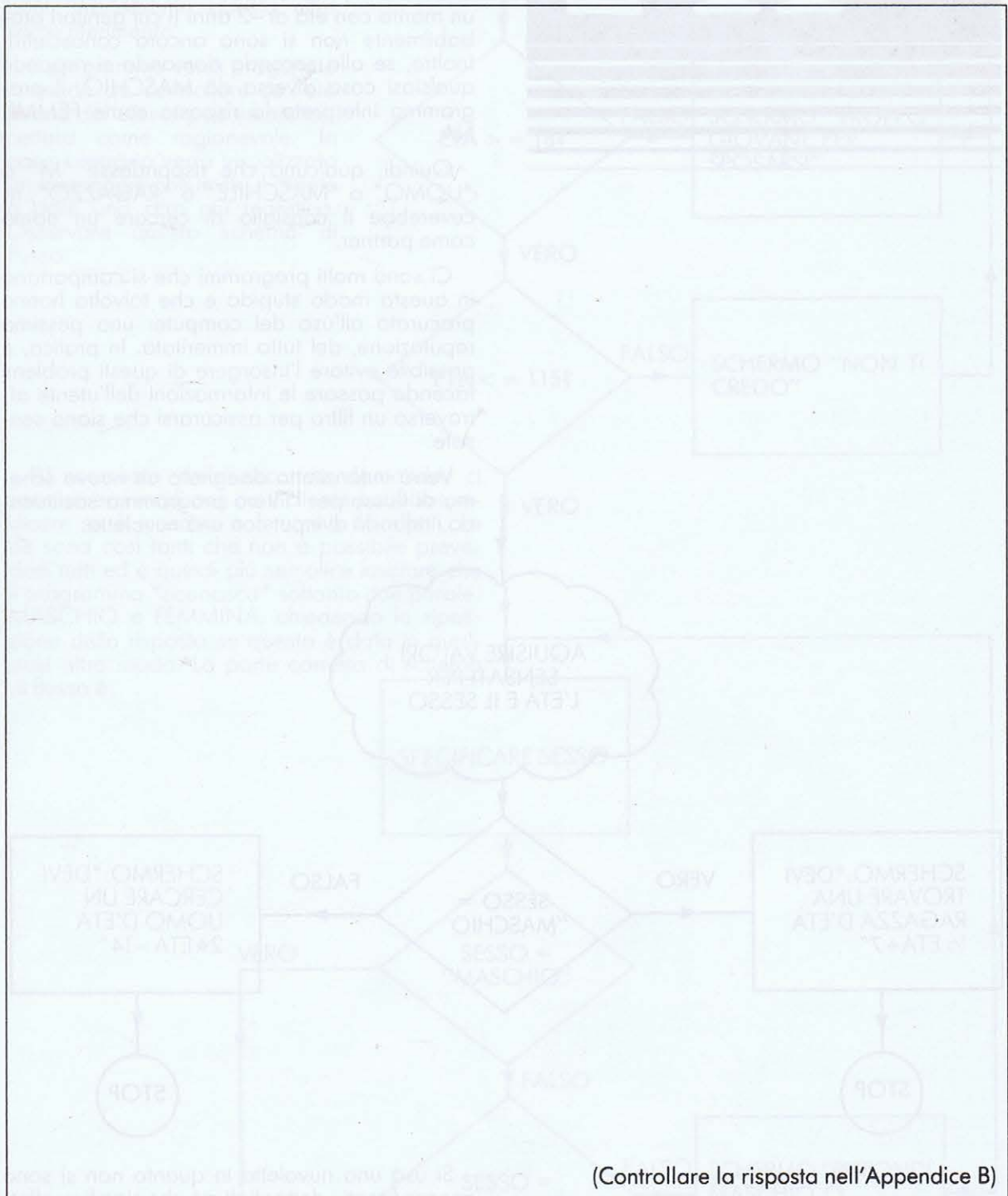
Per esempio, il programma

```
1Ø Q=1
2Ø PRINT Q;Q★Q
3Ø Q=Q+1
4Ø GOTO 2Ø
```

corrisponde a uno schema di flusso di questo tipo:

Ora disegnare uno schema di flusso per il seguente programma. Usare una mascherina di plastica per disegnare i blocchi:

```
10 S=1
20 PRINT S,12*S
30 S=S+1
40 IF S<13 THEN 20
50 STOP
```

(Controllare la risposta nell'Appendice B)

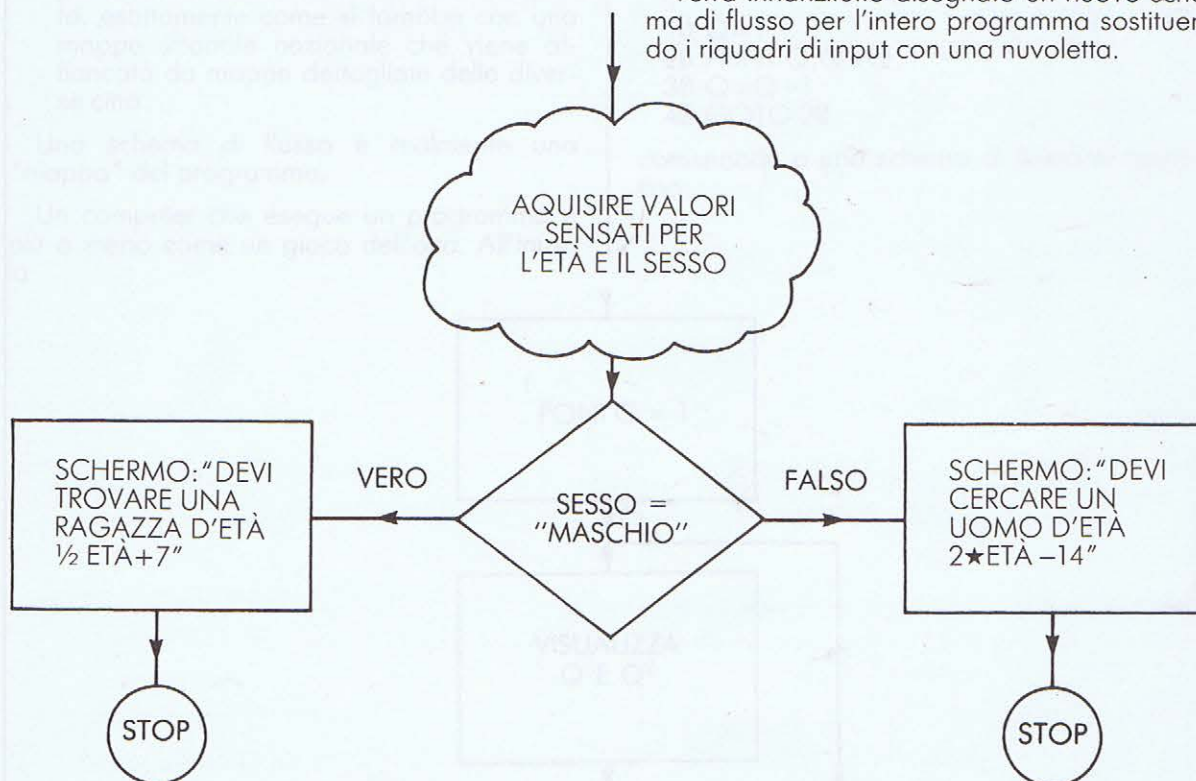
Esercizio 11.1 completato

ESERCIZIO 11·2

Verranno ora esaminate più dettagliatamente le tecniche fin qui descritte. Una caratteristica del programma per la guida al matrimonio consiste nel fatto che se riceve dati non corretti, dà risposte assurde. Ciò viene definito con la sigla "GIGO", che sta per "Garbage In, Garbage Out" (in altre parole "input senza valore, out put senza valore").

Per esempio, una ragazza che dichiarasse di avere 6 anni, riceverebbe il consiglio di cercare un marito con età di –2 anni (i cui genitori probabilmente non si sono ancora conosciuti!). Inoltre, se alla seconda domanda si risponde qualsiasi cosa diversa da MASCHIO, il programma interpreta la risposta come FEMMINA.

Quindi, qualcuno che rispondesse "M" o "UOMO" o "MASCHILE" o "RAGAZZO", riceverebbe il consiglio di cercare un uomo come partner.

Ci sono molti programmi che si comportano in questo modo stupido e che talvolta hanno procurato all'uso del computer una pessima reputazione, del tutto immeritata. In pratica, è possibile evitare l'insorgere di questi problemi facendo passare le informazioni dell'utente attraverso un filtro per assicurarsi che siano sensate.

Verrà innanzitutto disegnato un nuovo schema di flusso per l'intero programma sostituendo i riquadri di input con una nuvoletta.

Si usa una nuvoletta in quanto non si sono ancora fissati i dettagli di ciò che significa effettivamente "sensato". La nuvoletta consente di pianificare il programma come un'intera unità, ma comporta l'elaborazione dell'azione in maggior dettaglio. Finora la pianificazione non è completa, ma ciò non significa che lo schema di flusso generale sia inutile o sbagliato.

Cosa significa dunque "sensato?" Si consideri innanzitutto l'età dell'utente. Il valore inferiore probabile è 18 in quanto le persone al di sotto dei 18 anni non si recano spesso all'agenzia matrimoniale. Il limite superiore è difficile da stabilire, ma secondo il Guinness dei Primati la più vecchia persona vivente ha 115 anni. Prenderemo questa cifra come riferimento.

Il programma verrà progettato in modo da controllare se l'età fornita dal cliente può essere accettata come ragionevole. In caso contrario verrà visualizzato un messaggio che invita il cliente a dare una cifra più realistica. Osservare questo schema di flusso:

Al momento della seconda domanda, ci sono diversi modi con cui il cliente potrebbe indicare se è maschio o femmina. In effetti, ce ne sono così tanti che non è possibile prevederli tutti ed è quindi più semplice lasciare che il programma "riconosca" soltanto due parole: MASCHIO e FEMMINA, chiedendo la ripetizione della risposta se questa è data in qualsiasi altro modo. La parte corretta di schema di flusso è

SPECIFICARE SESSO

SESSO = "MASCHIO"

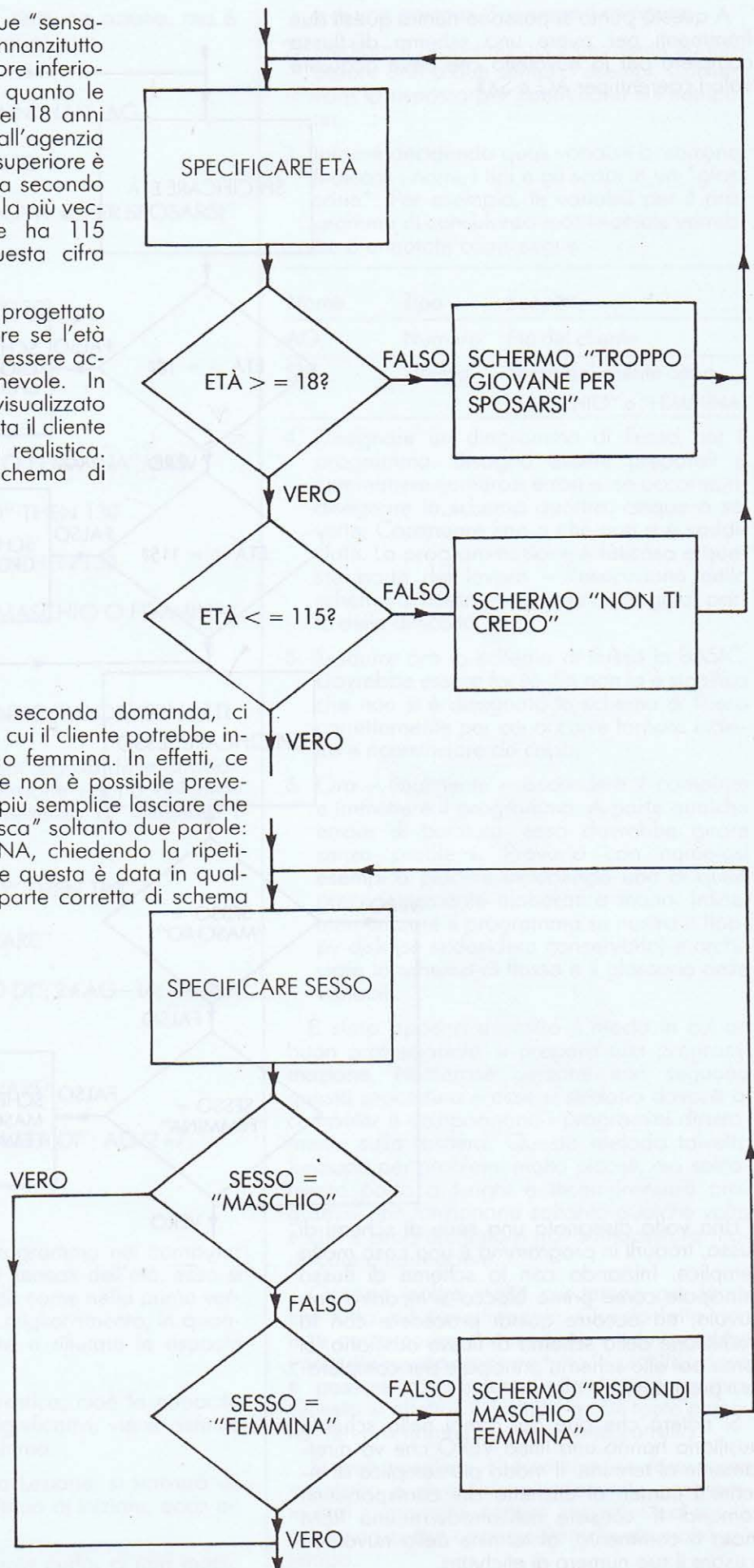
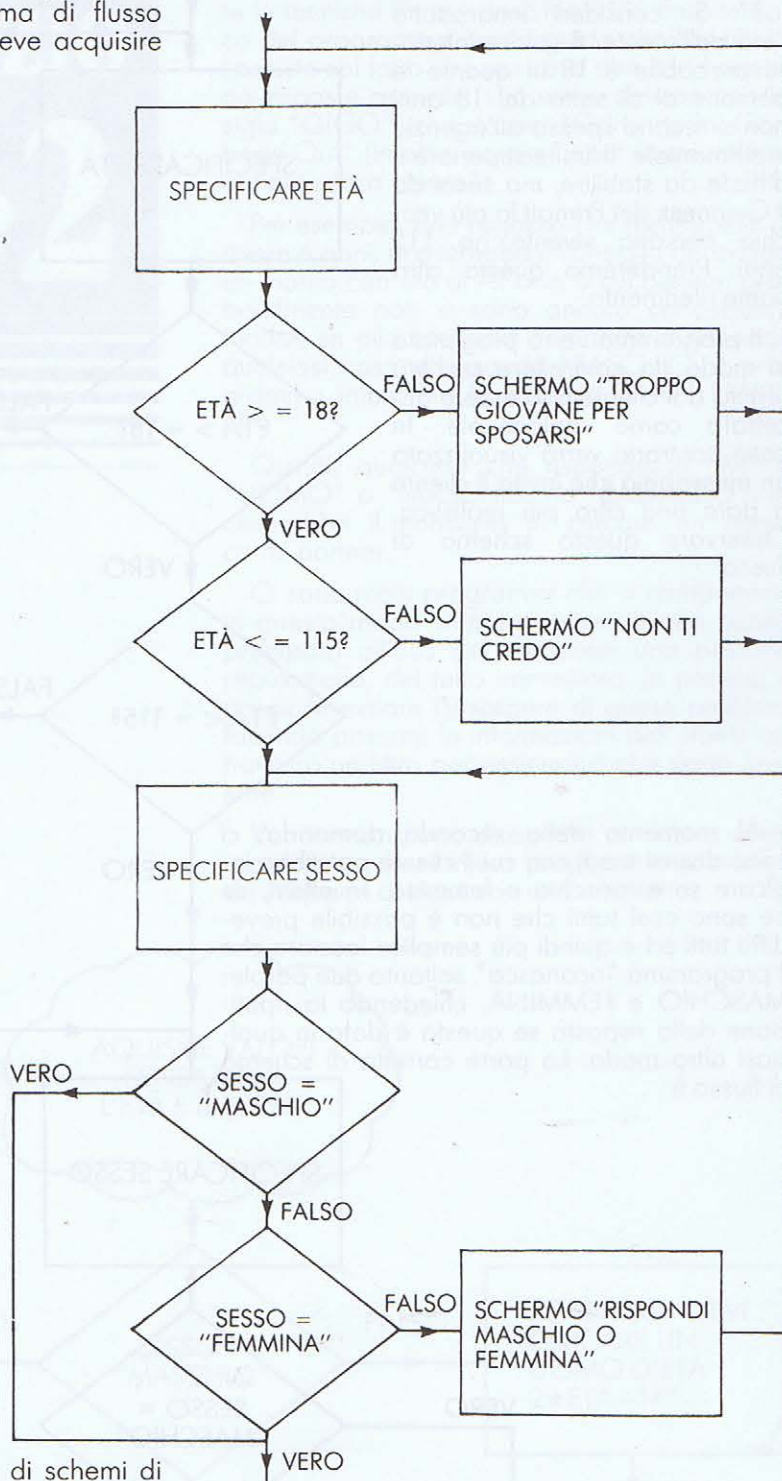
VERO

FALSO

SESSO = "FEMMINA"

FALSO

SCHERMO "RISPONDI MASCHIO O FEMMINA"

VERO

A questo punto si possono riunire questi due frammenti per avere uno schema di flusso completo per la nuvoletta che deve acquisire valori coerenti per AG e SX$.

Una volta disegnata una serie di schemi di flusso, tradurli in programma è una cosa molto semplice. Iniziando con lo schema di flusso principale come primo blocco si incontra una nuvola, ed occorre quindi procedere con la traduzione dello schema di flusso ausiliario. Si torna poi allo schema principale per completare il programma.

Si noterà che due dei rombi nello schema ausiliario hanno una linea VERO che va direttamente al termine. Il modo più semplice di inserire i numeri di etichetta dei corrispondenti comandi IF consiste nell'introdurre una REM (nota o commento) al termine della nuvoletta e usare il suo numero di etichetta.

La REM non compie nessuna azione, ma è un comodo punto di aggancio:

Si ottiene:

```
10 INPUT "QUANTI ANNI HAI"; AG
20 IF AG > = 18 THEN 50
30 PRINT "TROPPO GIOVANE PER SPOSARSI"
40 GOTO 10
50 IF AG < = 115 THEN 80
60 PRINT "NON TI CREDO"
70 GOTO 10
80 INPUT "MASCHIO O FEMMINA"; SX$
90 IF SX$="MASCHIO" THEN 130
100 IF SX$="FEMMINA" THEN 130
110 PRINT "RISPONDI MASCHIO O FEMMINA"
120 GOTO 80
130 REM AG E SX$ HANNO VALORI SENSATI
```

Questa parte viene poi seguita dalle istruzioni già scritte per la versione ridotta del programma (avendo l'avvertenza di sistemare i numeri di etichetta).

```
140 IF SX$="MASCHIO" THEN 180
150 PRINT "DEVI CERCARE"
160 PRINT "UN UOMO DI"; 2*AG-14; "ANNI"
170 STOP
180 PRINT "DEVI TROVARE"
190 PRINT "UNA RAGAZZA DI"; AG/2+7;
    "ANNI"
200 STOP
```

Immettere questo programma nel computer e provarlo. Per i valori sensati dell'età, esso si comporterà esattamente come nella prima versione, ma si noterà un miglioramento, in quanto verranno individuate e rifiutate le risposte sciocche.

Quest'ultima caratteristica, cioè la capacità di scartare dati non significativi, viene definita "solidità" di un programma.

Per terminare questa Lezione, si scriverà un proprio programma. Prima di iniziare, ecco alcuni punti consigliati:

1. Munirsi di abbondante carta, di una matita e di una gomma. Spegnere il computer.
2. Studiare il problema accuratamente ed elaborare uno o due semplici esempi. Conservare la risposta per controllarla sul computer.
3. Iniziare decidendo quali variabili occorrono. Indicare i nomi, i tipi e gli scopi in un "glossario". Per esempio, le variabili per il programma di consulenza matrimoniale verrebbero annotate come segue:

Nome	*Tipo*	*Scopo*
AG	Numero	Età del cliente
SX$	Stringa	Sesso del cliente ossia "MASCHIO" o "FEMMINA"

4. Disegnare un diagramma di flusso per il programma. Bisogna essere preparati a commettere numerosi errori e, se occorre, ridisegnare lo, schema quattro, cinque o sei volte. Continuare fino a che non si è soddisfatti. La programmazione è faticosa e questa parte del lavoro – l'esecuzione dello schema di flusso – contiene la maggior parte della difficoltà.
5. Tradurre ora lo schema di flusso in BASIC. Dovrebbe essere facile. Se non lo è significa che non si è disegnato lo schema di flusso correttamente per cui occorre tornare indietro e ricominciare da capo.
6. Ora – finalmente – accendere il computer e immettere il programma. A parte qualche errore di battitura, esso dovrebbe girare senza problemi. Provarlo con numerosi esempi a piacere includendo uno di quelli precedentemente elaborati a mano. Infine, memorizzare il programma su nastro o floppy disk (se si desidera conservarlo) e archiviare lo schema di flusso e il glossario delle variabili.

È stato appena descritto il modo in cui un buon professionista si prepara alla programmazione. Numerose persone non seguono questa procedura – esse si siedono davanti ai computer e compongono i programmi direttamente sulla tastiera. Questo metodo talvolta funziona per problemi molto piccoli, ma solitamente porta a lunghi e incomprensibili programmi che funzionano soltanto qualche volta e che il programmatore trova impossibile modificare o correggere.

Occorre inoltre molto più tempo per far sì che tutto giri correttamente. Questo fatto non è di comprensione immediata – sembrerebbe infatti più rapido ignorare tutta la pianificazione e passare direttamente alla fase esecutiva. Questo in effetti è il motivo per cui tante persone non programmano in maniera corretta.

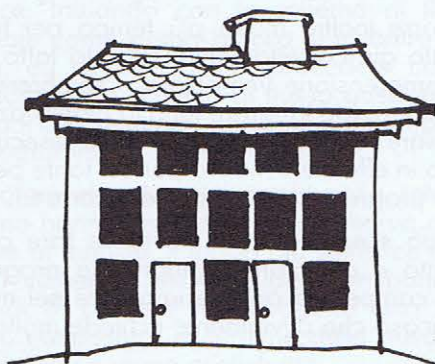
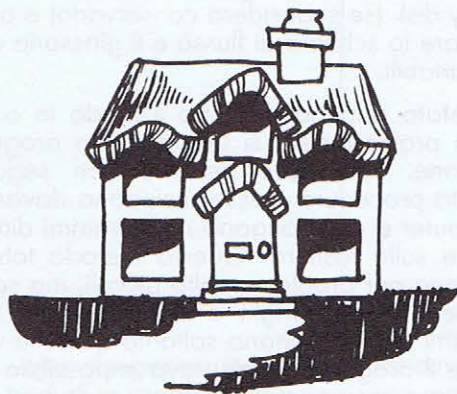
Si può scegliere liberamente se fare come suggerito e diventare rapidamente programmatori competenti oppure imparare nel modo più faticoso che ovviamente richiede molto più tempo.

Eseguire ora la pianificazione, lo schema di flusso, la stesura e la prova di un programma per il seguente problema:

In Ruritania, la tassa sulla casa è strutturata come segue:

Per ciascuna porta:	Lire 57
Per ciascuna finestra:	Lire 12
Per ciascun tetto coperto di paglia:	Lire 38
Per ciascun tetto coperto con tegole:	Lire 94

Supponendo che tutte le case debbano essere o ricoperte con paglia o ricoperte con tegole, scrivere un programma per chiedere i particolari di qualsiasi casa e visualizzare la tassa da pagare. Per esempio, la risposta esatta per un cottage con tetto di tegole, con una porta e due finestre sarebbe di Lire (38+57+2★12)=Lire 119.

Fare in modo che il programma visualizzi gli importi delle tasse per le seguenti case (supponendo che tutte le porte e le finestre si trovino nella parte anteriore):

Controllare le risposte nell'Appendice B.

Esercizio 11.2 completato

ESERCIZIO 11·3

Caricare ed eseguire il programma UNIT11PROG.

Dopo averlo listato, esaminare la sequenza delle istruzioni e disegnare lo schema di flusso e il glossario relativi.

Esercizio 11.3 completato	

LEZIONE: 12

Non occorre cercare molto a lungo in un programma per trovarvi un'iterazione in qualche punto. Le iterazioni sono così comuni e importanti che il linguaggio BASIC fornisce un metodo per concentrare in due comandi i quattro elementi che, come si ricorderà, sono:

- La scelta della variabile di controllo
- Il valore iniziale per la variabile di controllo
- Il valore finale per la variabile di controllo
- L'incremento o quantità di cui la variabile cresce ogni volta nel corso dell'iterazione.

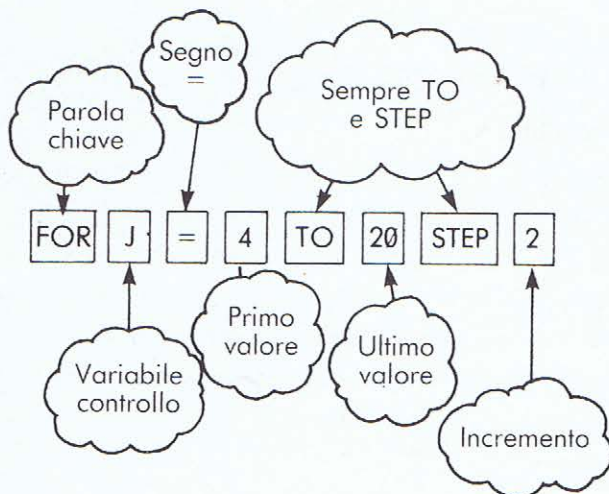
Tutte queste parti possono essere riunite in uno speciale comando che usa la parola chiave FOR, che è tutto ciò che occorre per impostare un'iterazione oltre al comando NEXT per contrassegnare la fine del corpo dell'iterazione.

Confrontare i seguenti due programmi, che danno esattamente lo stesso risultato:

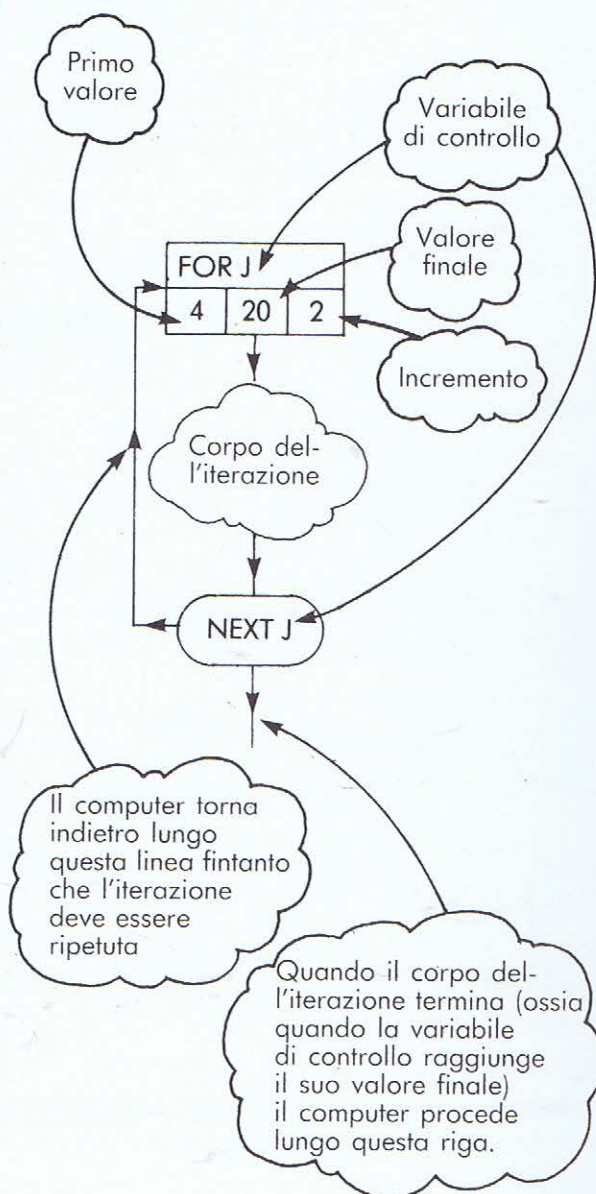
1Ø J=4	1Ø FOR J=4 TO 2Ø STEP 2
2Ø PRINT J,J★7	2Ø PRINT J,J★7
3Ø J=J+2	3Ø NEXT J
4Ø IF J<22 THEN 2Ø	
(Usando IF .... THEN)	(Usando FOR .... NEXT)

In entrambi i casi

La variabile di controllo è	J
Il primo valore è	4
L'ultimo valore è	2Ø
L'incremento è	2

L'esempio mostra come è costruito il comando FOR

Il comando NEXT cita il nome della variabile di controllo come riscontro per aiutare a leggere il programma.

Ogni comando FOR deve avere un corrispondente NEXT e tra di essi viene racchiuso il corpo del loop.

Negli schemi di flusso vengono mostrate le iterazioni in un modo speciale usando blocchi che non possono essere facilmente confusi con altri tipi di azione:

ESERCIZIO 12.1

Per facilitare la comprensione e l'apprendimento del comando FOR, osservare i seguenti brevi programmi e scrivere la propria previsione su ciò che il computer visualizzerà. Controllare quindi la risposta sulla macchina:

(i)
```
10 FOR Q=1 TO 16 STEP 5
20 PRINT Q;
30 NEXT Q
40 STOP
```

Previsione:

(ii)
```
10 FOR R=38 TO 50 STEP 3
20 PRINT R; 50-R
30 NEXT R
40 STOP
```

Previsione:

Tradurre ora il seguente programma utilizzando una notazione FOR-NEXT. Controllare la risposta eseguendo entrambe le versioni sul computer e assicurandosi che diano le stesse risposte:

```
10 PRINT "TABELLINA DEL NOVE"
20 S=1
30 PRINT S; "PER 9 =";
40 PRINT 9★S
50 S=S+1
60 IF S<13 THEN 30
70 STOP
```

Traduzione:

Ci sono alcuni punti che occorre ricordare a proposito dei comandi FOR e NEXT:

(a) Se l'incremento o dimensione del passo è 1, si può evitare di scrivere "STEP 1" al termine del comando FOR, in quanto in assenza di precisazioni il computer utilizza il valore 1.

(b) Il controllo del loop può essere realizzato con un "conteggio alla rovescia" usando un valore di incremento negativo.
Il programma

```
10 FOR X=10 TO 5 STEP -1
20 PRINT X;
30 NEXT X
```

visualizzerà: `10 9 8 7 6 5`

nell'ordine.

(c) Il corpo dell'iterazione è sempre eseguito almeno una volta anche se il valore finale è minore del valore iniziale. Per esempio:

```
10 FOR R=5 TO 3
20 PRINT R
30 NEXT R
```

visualizzerà `5`

(d) I valori contenuti nel comando FOR non devono essere necessariamente numeri ma

possono essere espressioni che comprendono altre variabili.

Per esempio, il seguente programma visualizzerà il numero dei simboli "cuore" richiesti dall'utente. Provarlo e studiarlo accuratamente:

```
10 INPUT "QUANTI CUORI"; H
20 FOR K=1 TO H
30 PRINT " [CTRL] e [3 RED] ♥";
40 NEXT K
50 STOP
```


(e) La variabile di controllo non può essere una stringa. Per esempio, il "comando"

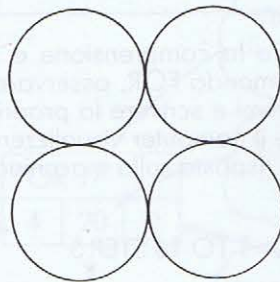
darebbe un SYNTAX ERROR e non è pertanto ammesso usare questa costruzione. Tenendo conto di ciò, prevedere il risultato dei seguenti programmi e controllare i risultati sul computer:

(i)
```
10 FOR A=1 TO 4
20 PRINT A★A;
30 NEXT A
40 STOP
```

(ii)
```
10 FOR B=3 TO 0 STEP -1
20 PRINT B;
30 NEXT B
40 STOP
```

(iii)
```
10 FOR C=5 TO 4
20 PRINT C;
30 NEXT C
40 STOP
```

(iv)
```
10 X=5
20 Y=9
30 Z=2
40 FOR W=X TO Y STEP Z
50 PRINT W;
60 NEXT W
70 STOP
```

Finora ci si è concentrati sui dettagli dei comandi FOR e NEXT, scegliendo accuratamente i corpi delle iterazioni che vengono controllate in modo da farli risultare i più semplici possibili. In pratica il corpo di un loop non deve essere necessariamente corto e semplice, ma può essere complesso a piacere – ricordando però che viene eseguito ogni volta che il computer esegue l'iterazione stessa. Supponiamo che venga chiesto di costruire una piramide a base quadrata costituita da palle di cannone. Si numereranno gli strati 1, 2, 3... iniziando dalla cima. Lo strato 1 sarà la punta, per la quale occorrerà soltanto una palla di cannone. Lo strato 2 sarà il secondo e occorreranno 4 palle disposte come segue:

Lo strato 3 richiederà 9 palle, lo strato 4 – 16 e così via.

Chiaramente il numero di palle da cannone che occorre per l'intera piramide dipende da quanti strati si intendono costruire. Una piramide a tre strati ha bisogno di 1+4+9 ossia 14 palle da cannone; una con 4 strati ne richiederà 1+4+9+16 ossia 30.

Se si ha in programma di costruire una piramide molto ampia, queste somme risulteranno piuttosto lunghe e noiose e si potrebbe pertanto utilizzare un programma di computer per eseguirle. Questo programma risponde alla domanda "quante palle da cannone occorrono per una piramide con 'N' strati?".

Nel disegnare il programma, un fattore chiave è il numero di palle da cannone per ciascun strato. I numeri 1, 4, 9 e 16... sono abbastanza familiari e in effetti di può notare che il numero delle palle di cannone di ciascun strato è il quadrato del numero che contraddistingue lo strato. Per esempio, lo strato 7 avrà bisogno di 7x7 ossia 49 palle da cannone.

Ora i dettagli per il programma. Si inizierà pensando alle variabili necessarie.

Il piano globale prenderà in considerazione gli strati uno per uno. Si farà cioè in modo che il computer calcoli quante palle occorrono per ogni strato e si sommerà questo numero ad un "totale mobile". All'inizio, il totale mobile deve essere posto a zero. Al termine, quando sono stati presi in considerazine tutti gli strati, il totale mobile indicherà il numero di palle da cannone occorrenti per costruire l'intera piramide. Questa è la risposta al problema.

Un nome adatto per il totale mobile è RT (Running Total).

Occorrono oltre due variabili:

(I) Il numero di strati nella piramide. Ricordare che il programmatore non conosce questo numero; tocca all'utente fornire qualsiasi valore desiderato. Un buon nome per questa variabile è L (Layers).

(II) Quando il programma viene eseguito, prenderà in considerazione prima lo strato 1 quindi lo strato 2, lo strato 3 e così via. Occorre una variabile per indicare con quale dei diversi strati L il programma sta operando in quel momento. Un nome adatto di variabile è V. V assumerà tutti i valori compresi tra 1 e L (il numero dello strato inferiore). Costruirà quindi ovviamente la variabile di controllo di un comando FOR, pertanto:

```
FOR V=1 TO L
NEXT V
```

Il glossario relativo al programma è di conseguenza:

Nome	*Scopo*
RT	Per conservare il totale mobile delle palle da cannone
L	Numero degli strati nella piramide
V	Numero dello strato col quale il programma opera in qualsiasi momento

Successivamente si scriveranno alcune delle azioni che il programma deve intraprendere:

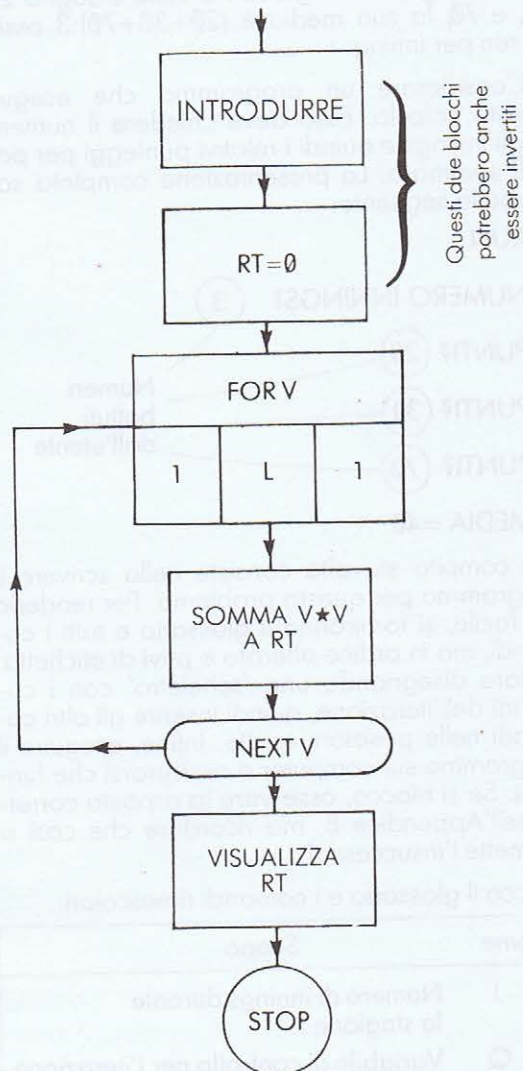
Aggiungere a RT V elevato al quadrato	(Ciò somma il numero delle palle da cannone per lo strato numero V)
Stampare RT	(Visualizza i risultati)
Impostare RT=Ø	(Azzera RT)
Immettere L	(Chiede all'utente quanti strati ci sono nella sua piramide)
FOR V=1 TO L NEXT V	(Controllo dell'iterazione che conteggia gli strati)
STOP	

Questi sono i frammenti del programma che si sta sviluppando, che devono però essere messi insieme nell'ordine giusto. Si era già previsto l'uso di loop e sarà di grande aiuto essere in grado di stabilire, per ciascun comando, se deve essere eseguito

prima dell'inizio dell'iterazione

o all'interno dell'iterazione (come parte del corpo dell'iterazione)

o dopo la fine dell'iterazione.

Si possono usare vari segnali. Prima che l'iterazione possa iniziare. Il programma deve sapere quante volte eseguire l'iterazione stessa, così l'introduzione del valore di L deve venire prima dell'iterazione. La stessa cosa vale per il comando che imposta RT o zero. Il numero totale di palle da cannone per l'intera piramide ne comprende sempre alcune per ciascuno strato. Il comando per aggiungere il valore di uno strato a RT deve essere ripetuto molte volte per cui va posto all'interno dell'iterazione. Infine, il computer non può fornire la risposta esatta fino a che non ha preso in considerazione tutti gli strati, per cui il comando PRINT può venire usato solo dopo che l'iterazione è terminata. A questo punto è possibile disegnare uno schema di flusso:

E il corrispondente programma è

```
10 INPUT "NUMERO STRATI"; L
20 RT=Ø
30 FOR V=1 TO L
40 RT=RT+V★V
50 NEXT V
60 PRINT RT; "PALLE CANNONE RICHIESTE"
70 STOP
```

Immettere questo programma e provarlo.

Ora un altro problema. Nel gioco del cricket, un giocatore può avere un numero di "innings" o turni durante la stagione. Ogni volta egli totalizza qualche "run" o corsa: molte se è un giocatore valido o fortunato, poche (o addirittura nessuna) se non è nè valido nè fortunato. Se si vuole conoscere come un giocatore si è comportato per l'intera stagione, è possibile calcolare il numero medio di "run" per ogni "inning". Ciò si ottiene sommando tutti i run acquisiti durante la stagione e dividendo il risultato per il numero degli innings. Per esempio, se il giocatore gioca tre volte e segna 2Ø, 3Ø, e 7Ø, la sua media è (2Ø+3Ø+7Ø):3 ossia 4Ø run per inning.

Considerare un programma che esegue questo calcolo. Esso deve chiedere il numero degli innings e quindi i relativi punteggi per poterli sommare. La presentazione completa sarebbe la seguente:

Il compito stavolta consiste nello scrivere il programma per questo problema. Per renderlo più facile, si forniranno il glossario e tutti i comandi, ma in ordine alterato e privi di etichetta. Iniziare disegnando uno 'scheletro' con i comandi dell'iterazione, quindi inserire gli altri comandi nelle posizioni esatte. Infine, eseguire il programma sul computer e assicurarsi che funzioni. Se si blocca, osservare la risposta corretta nell'Appendice B, ma ricordare che così si ammette l'insuccesso!

Ecco il glossario e i comandi rimescolati:

Nome	*Scopo*
J	Numero di innings durante la stagione
Q	Variabile di controllo per l'iterazione
RS	Usato per sommare i run totali segnati
S	Punteggio per ciascun inning

```
NEXT Q
INPUT"NUMERO INNINGS";J
INPUT"PUNTI";S
PRINT"MEDIA=";RS/J
STOP
RS=Ø
FOR Q=1 TO J
RS=RS+S
```

Esercizio 12.1 completato	

ESERCIZIO 12·2

Questa sezione terminerà con un problema da risolvere senza alcun aiuto. Se si va all'ufficio postale è facile rimanere bloccati in una coda perché qualcuno deve comprare una grande quantità di francobolli. Lo si sente dire all'impiegato dello sportello:

"ottantatre da 11 lire

e centodiciassette da 14 lire

e trentacinque da 75 lire"

e così via. Quando tutti i francobolli sono stati contati, all'impiegato serve un tempo notevole per calcolarne il costo totale.

Scrivere un programma per aiutare l'impiegato. La visualizzazione dovrebbe apparire più o meno come questa:

RUN

NUMERO BLOCCHI?

BLOCCO 1

NUMERO FRANCOBOLLI?

VALORE UNITARIO?

BLOCCO 2

NUMERO FRANCOBOLLI?

VALORE UNITARIO?

BLOCCO 3

NUMERO FRANCOBOLLI?

VALORE UNITARIO?

BLOCCO 4

NUMERO FRANCOBOLLI?

VALORE UNITARIO?

TOTALE=2Ø79 LIRE

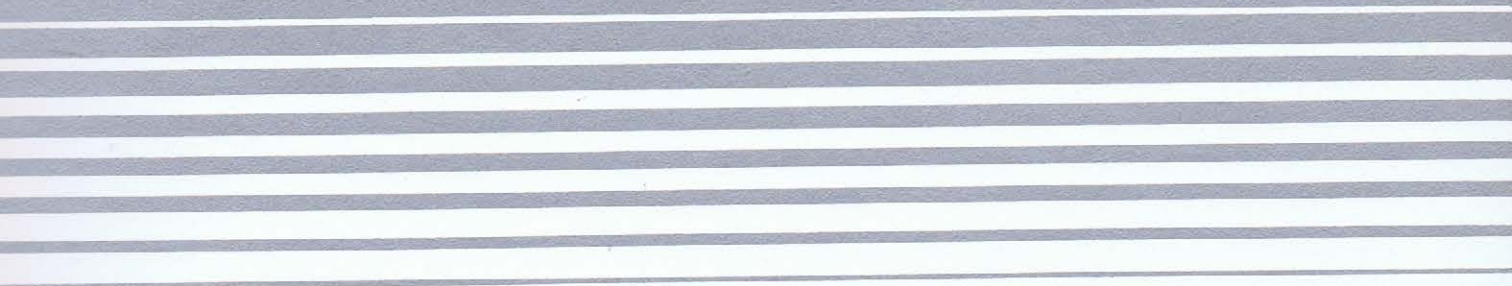
Il programma dovrebbe contenere 1Ø comandi (compreso lo STOP). Quattro di questi comandi formeranno il corpo di un'iterazione, eseguita una volta per ogni blocco. In ogni caso non tentare di scrivere il programma senza avere eseguito il relativo progetto con glossario e schema di flusso. Prevedere un tempo sufficientemente lungo.

Se nonostante lo spazio impiegato non si è in grado di risolvere il problema correttamente, studiare nuovamente le ultime lezioni per consolidare la propria conoscenza.

Confrontare infine la risposta con quella indicata nell'appendice B.

Esercizio 12.2 completato	

Il quiz di auto-test per la Lezione 12 è "UNIT12QUIZ".

LEZIONE: 13

Questa Lezione riguarda un argomento facile e divertente: l'uso del computer per creare suoni e note musicali.

Caricare il programma intitolato SOUND DEMO, alzare il volume dell'apparecchio televisivo e provare la selezione di effetti sonori del programma; questo mette a disposizione una gamma di suoni piuttosto ampia, consentendo di avere un'idea delle possibilità del computer in questo campo.

Probabilmente ben presto si desidererà progettare e programmare suoni di propria creazione, e il modo in cui si può eseguire tutto ciò costituisce l'argomento di questa Lezione.

La produzione di suoni sul computer viene controllata da due comandi: VOL e SOUND.

Per prima cosa, verrà illustrato VOL, dal momento che è il più semplice dei due. In un programma, la parola chiave VOL viene sempre seguita da un numero (o da un'espressione) il cui valore è compreso tra Ø e 8. Il comando controlla l'intensità del suono che il computer crea; in altre parole, costituisce una vera e propria "regolazione di volume". L'intensità massima del suono corrisponde a

VOL 8

mentre si ha il silenzio totale con

VOL Ø

Livelli intermedi del volume vengono selezionati da VOL 2, VOL 4, e così via.

Quando si scrive un programma che prevede dei suoni, si dovrebbe sempre inserire un comando

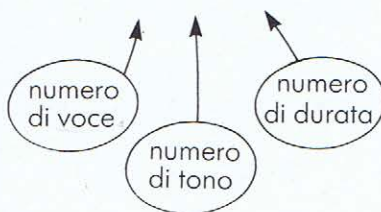
VOL 7

nella parte iniziale del programma stesso. Quando il programma viene eseguito, il controllo del volume sull'apparecchio televisivo (o sullo schermo video) deve essere alzato, in caso contrario non è possibile udire i suoni prodotti.

Il comando utilizzato per produrre effettivamente delle note musicali è SOUND. Ogni nota possiede due importanti caratteristiche:

- il tono (alto o basso)
- la durata (quanto a lungo viene mantenuta la nota)

In un comando SOUND, la parola chiave è seguita da tre numeri o espressioni, come nell'esempio che segue:

SOUND 1,8ØØ,2Ø

Il computer ha tre tipi diversi di voce, che corrispondono ai numeri 1, 2 e 3. Per ora verrà usata solo la voce numero 1. Tutti i comandi SOUND inizieranno

SOUND 1,......

Il tono della nota viene controllato dal numero di tono, che può avere un valore compreso fra Ø e 1Ø23. In altre parole, maggiore è il valore del numero di tono, più alta è la nota, ma questa corrispondenza non è direttamente proporzionale.

Il breve programma che segue fa in modo che il computer emetta tutti i toni possibili compresi fra Ø e 1Ø23, e consente di farsi un'idea dei limiti della gamma vocale del computer stesso:

```
10 VOL7
20 FOR J=0TO 1023
30 SOUND1,J,1
40 NEXT J
```

Il diagramma seguente mostra la relazione che intercorre fra i numeri di tono e le note della scala musicale. Ciò sarà utile al momento di scrivere della musica per il computer.

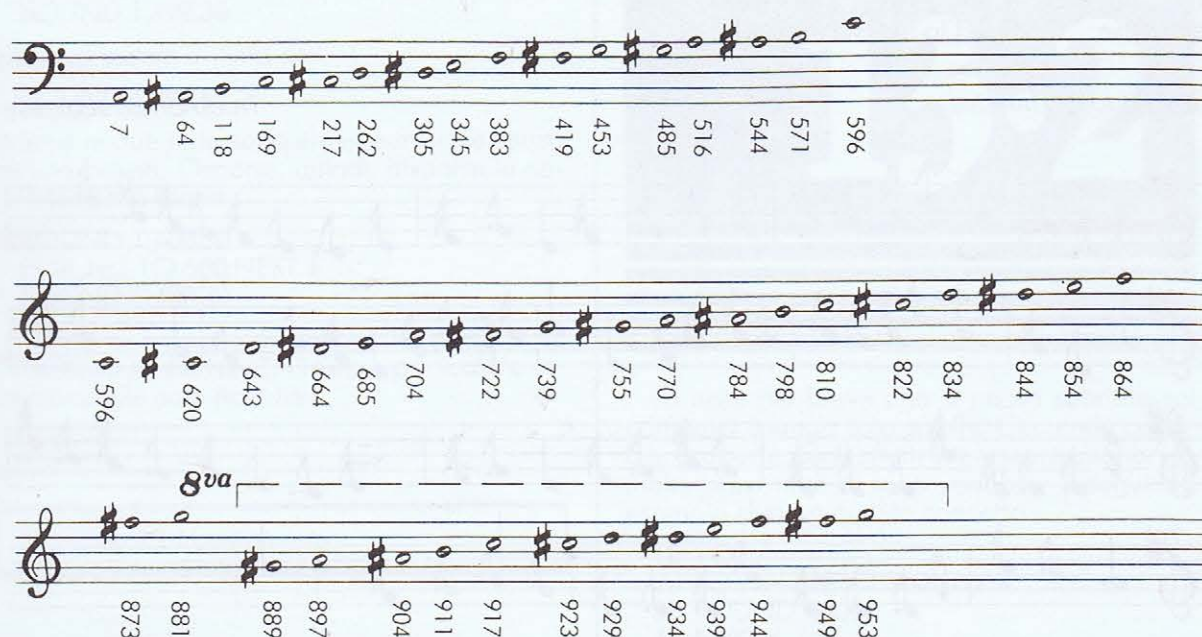



Figura 13.1

La durata della nota, espressa in "jiffy" (1 jiffy equivale a 1/6Ø di secondo), è impostata dal numero di durata. Per esempio, la nota da:

SOUND 1,9ØØ,12Ø

durerà esattamente 12Ø jiffy, equivalenti a due secondi.

Uno degli usi più semplici e divertenti della funzione sonora è quello di informare l'utente se la risposta data a un certo messaggio è accettabile. Considerare un programma nel quale il computer domanda all'utente se preferisce vino rosso (R), vino bianco (W), birra (B) o niente (N). L'utente viene invitato a battere la lettera iniziale indicante la sua scelta. Se batte R, W, B o N (come previsto dal programma) il computer emette un suono acuto e melodioso, per simulare approvazione, quindi passa alla domanda successiva. Se l'utente batte qualcosa di diverso dalle lettere indicate, il computer emette un suono sordo e ripete il messaggio. Il codice per questa sequenza è il seguente:

```
 1Ø VOL 7
 2Ø PRINT"VINO ROSSO (R),
    VINO BIANCO (W)"
 3Ø PRINT"BIRRA (B) O NIENTE (N)"
 4Ø INPUT"INTRODURRE CARATTERE
    CORRISPONDENTE";L$
 5Ø IF L$="R" THEN 14Ø
 6Ø IF L$="W" THEN 14Ø
 7Ø IF L$="B" THEN 14Ø
 8Ø IF L$="N" THEN 14Ø
 9Ø PRINT
1ØØ PRINT"SI PREGA RISPONDERE
    R,W,B O N"
11Ø PRINT
12Ø SOUND 1,2Ø,7Ø:REM TONO BASSO
13Ø GOTO 2Ø
14Ø SOUND 1,93Ø,12:REM TONO ACUTO
15Ø PRINT"OK"
16Ø ...
```

Un altro utilizzo del comando SOUND è fare in modo che il computer suoni dei motivi. Ogni nota di una melodia può essere suonata dal proprio comando SOUND. Qui di seguito viene data, a titolo di esempio, una celebre aria scozzese, "The Road to the Isles".

The Road to The Isles *Tradizionale scozzese*

Figura 13.2

La prima parte della melodia è stata codificata nel programma TUNE, che può essere caricato ed eseguito. Quando si lista il programma, si osserverà come è stata eseguita la codifica. Ogni nota viene tradotta in un comando SOUND. Ogni battuta del motivo è lunga 32 jiffy (ciò corrisponde ad una velocità di metronomo di 60★60/32 o 112 colpi al minuto). La prima nota è una croma punteggiata, o "tre quarti", accordata sul re, codificata come

SOUND 1,643,24

tono per il re — tre quarti di 32

Uno degli accorgimenti più usati nel campo della composizione musicale è la pausa. Generalmente, è possibile fare in modo che il computer compia una pausa, per qualsiasi durata di tempo, fornendogli un "loop" vuoto, come questo:

```
FOR J=1 TO 1000:NEXT J
```

Però, per quanto riguarda la musica, la procedura non è così semplice. Cancellare la memoria e battere:

```
10 VOL 7
20 SOUND 1,400,120
30 FOR J=1 TO 1000:NEXT J
40 SOUND 1,400,120
```

Al momento dell'esecuzione di questo programma, però, ci si accorge che la pausa che ci si aspetterebbe di trovare tra le due note non è presente. Ciò accade poiché il formato SOUND dà semplicemente avvio a una nota. A quel punto la "voce" del computer continua l'emissione della nota, mentre il computer stesso procede nell'esecuzione del programma.

In altre parole, il computer viene ritardato solo quando tenta di emettere un suono mentre la nota precedente non è ancora conclusa.

A questo punto, è possibile capire che cosa ha impedito al computer di rispettare la pausa prescritta dal programma. Dopo aver iniziato l'emissione della prima nota, il programma è subito passato al "loop" vuoto della riga 30. Dopo un secondo ha completato il "loop" e ha quindi tentato di emettere la seconda nota, ma la prima non era ancora conclusa. Al termine della prima nota, è stata quindi emessa immediatamente la seconda, senza nessuna pausa.

A questo punto, la soluzione a questo problema dovrebbe essere chiara. Se si desidera che una nota venga seguita da una pausa, il "loop" vuoto deve poter assorbire tutto il tempo della durata della nota, oltre al tempo della pausa. Ad esempio, se si desidera che una nota di un secondo sia seguita da una pausa, sempre di un secondo, il "loop" deve durare due secondi.

Un problema analogo sorge se si desiderano suonare due note dello stesso tono. La sequenza

```
SOUND 1,400,30
SOUND 1,400,30
```

non può essere distinta da

```
SOUND 1,400,60
```

poiché le due note sono emesse insieme senza alcuna pausa. Occorre, quindi, disporre la sequenza nella forma:

```
SOUND 1,400,30
FOR J=1 TO 500:NEXT J
SOUND 1,400,30
```

Come Esercizio, completare la codifica di "The Road to the Isles". Prestare particolare attenzione alle note ripetute.

Esercizio 13.1 completato	

ESERCIZIO 13·2

La nota più breve che si possa suonare sul computer è lunga solo un jiffy. Ciò rende possibile produrre degli effetti interessanti suonando molte brevi note di tono variabile. Il seguente esempio chiarirà questo concetto:

```
10 VOL 7
20 FOR X=700 TO 900 STEP 4
30 SOUND 1,X,1
40 NEXT X
50 GOTO 20
```

Per ottenere un effetto diverso, provare ad aggiungere

```
35 SOUND 1,X+50,1
```

A questo punto verranno prese in esame le altre due "voci" del computer. La voce 2 da sola funziona nello stesso modo della voce 1. Se si prova ad usare entrambe le voci contemporaneamente, subentreranno notevoli interferenze elettroniche fra di esse; l'effetto può essere interessante, ma in ogni caso non è armonioso. La "voce" 3 viene utilizzata per produrre un tipo di suono chiamato generalmente "rumore bianco" - un sibilo con una scarsa definizione di tono. Questo tipo di suono è l'ideale per imitare il rumore di motori a vapore, aerei a reazione, razzi e astronavi. Provare

```
10 VOL 7
20 FOR X=850 TO 600 STEP -1
30 SOUND 3,X,2
40 NEXT X
```

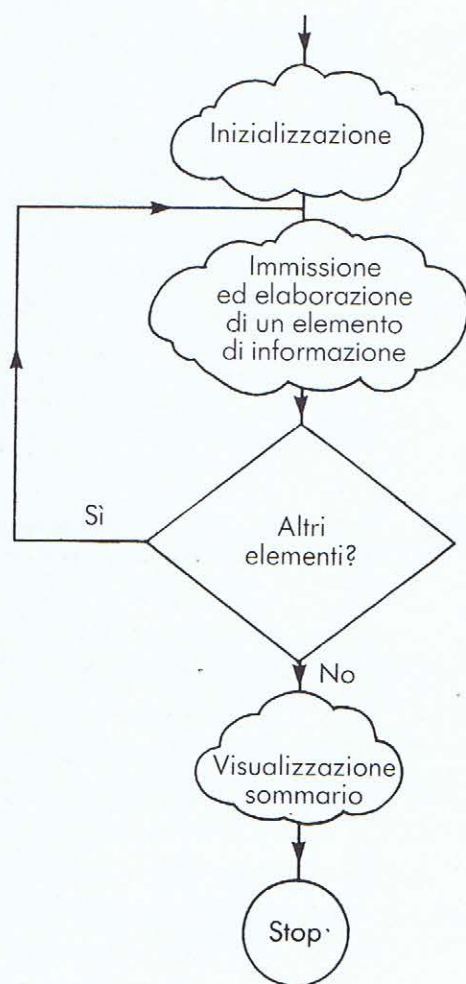
e il seguente esempio, ancora più completo

```
10 VOL 7
20 J=50
30 FOR L=1 TO 100
40 J=0.97*3
50 M=J+15
60 SOUND 3,860-M,M/3
70 FOR K= TO 4*M:NEXT K
80 NEXT L
```

Esercizio 13.2 completato	

LEZIONE: 14

In questa Lezione si esaminerà l'utilizzo del computer estremamente comune: un'applicazione in cui si fa in modo che la macchina immagazzini ed elabori una grande quantità di elementi separati d'informazione e visualizzi un sommario dei risultati. Per esempio, se si volesse tenere un resoconto aggiornato del proprio conto bancario, si potrebbe introdurre gli estremi di ogni assegno emesso e di ogni credito pagato alla banca e la macchina potrebbe dare il saldo al termine della settimana. Per fare un altro esempio, l'insegnante potrebbe immettere nel computer i voti degli esami dei vari allievi e il computer potrebbe presentare la media totale.

Tutti i programmi di questo tipo si adeguano allo stesso profilo base, che prevede uno schema di flusso più o meno di questo tipo:

Un esempio molto semplice è dato da questo programma che acquisisce 10 numeri e ne calcola la media:

```
10 S=0              } Inizializza
20 P=1
30 INPUT X          } Legge ed elabora
40 S=S+X              un elemento
50 P=P+1            } Altri elementi?
60 IF P < 11 THEN 30
70 PRINT"MEDIA="; S/10  } Visualizza
                          il sommario
80 STOP
```

Glossario

S: Usato per sommare i valori degli elementi
P: Usato per contare gli elementi
X: Usato per immettere i singoli elementi

Se non si comprende come funziona questo programma, controllarlo con i valori di input 3,6,2,7,0,9,8,3,12,10.

In questo esempio è stata usata una struttura IF-THEN per il controllo dell'iterazione, per rendere più chiara la costruzione del programma. In pratica, si scriverà il programma utilizzando una struttura FOR . . . NEXT, del tipo:

```
10 S = 0
20 FOR P = 1 TO 10
30 INPUT X
40 S = S+X
50 NEXT P
60 PRINT"LA MEDIA È"; S/10
70 STOP
```

Si esaminerà ora la parte del programma denominata "altri elementi". Nel primo esempio alla domanda veniva risposto tenendo un semplice conteggio e usando la condizione di P<11, che era vera fino a che il decimo non veniva immesso e sommato al totale mobile. Questo metodo dipende dal fatto che il programmatore conosca in anticipo quanti elementi devono entrare nel calcolo. Il metodo è pressochè inutile in pratica in quanto non è flessibile: occorrerebbero cioè diversi programmi per trovare la media di una serie di numeri composta da 11, 20 o qualsiasi altro numero di elementi.

È possibile scrivere un programma molto migliore se si suppone che l'utente possa comunicare al computer quanti elementi deve aspettarsi. Il seguente programma funzionerà per qualsiasi numero di elementi:

```
10 S = 0
20 INPUT"QUANTI NUMERI"; N
30 FOR P = 1 TO N
40 INPUT X
50 S = S + X
60 NEXT P
70 PRINT "LA MEDIA È"; S/N
80 STOP
```

Notare l'uso di N invece di 10

Glossario

S: È usato per sommare il valore degli elementi

P: È usato per contare gli elementi

X: È usato per immettere i singoli elementi

N: È usato per contenere il *numero* degli elementi

Finora tutto si è mosso su un terreno familiare, ma cosa succede quando l'utente deve immettere un grande numero di elementi (ad esempio un migliaio o più?). È illogico fargli contare gli elementi in anticipo, non è realistico supporre che egli inserisca il numero esatto.

Un modo per controllare una iterazione consiste nel comunicare semplicemente al computer che è terminato il flusso degli elementi senza usare un conteggio predeterminato. Si potrebbe ad esempio fare in modo che l'utente risponda alla domanda "altri elementi" ogni volta che viene eseguita l'iterazione. Ciò darebbe luogo ad un programma del tipo:

```
10 S = 0
20 N = 0
30 INPUT "PROSSIMO NUMERO"; X
40 S = S + X
50 N = N + 1
60 INPUT "ALTRI NUMERI"; M$
70 IF M$ = "SI" THEN 30
80 PRINT "LA MEDIA È"; S/N
90 STOP
```

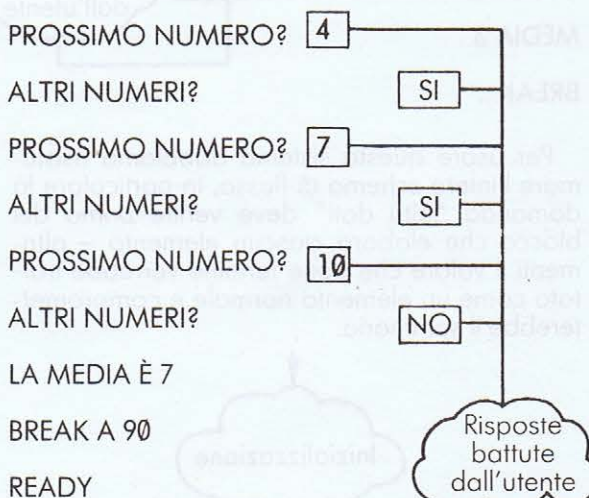
Glossario:

S: È usato per sommare il valore degli elementi

N: È usato per contare gli elementi

X: È usato per immettere i singoli elementi

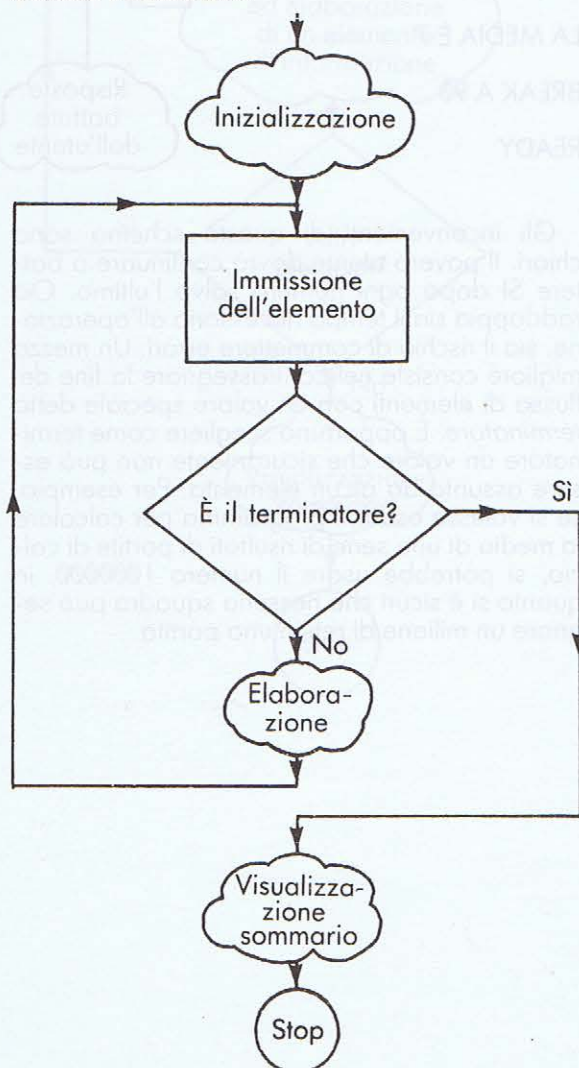
M$: È usato per contenere la risposta alla domanda "Altri elementi"

Se si eseguisse questo programma la visualizzazione potrebbe essere del tipo:

Gli inconvenienti di questo schema sono chiari. Il povero utente dovrà continuare a battere SI dopo ogni numero salvo l'ultimo. Ciò raddoppia sia il tempo necessario all'operazione, sia il rischio di commettere errori. Un mezzo migliore consiste nel contrassegnare la fine del flusso di elementi con un valore speciale detto *terminatore*. È opportuno scegliere come terminatore un valore che sicuramente non può essere assunto da alcun elemento. Per esempio, se si volesse usare il programma per calcolare la media di una serie di risultati di partite di calcio, si potrebbe usare il numero 1000000, in quanto si è sicuri che nessuna squadra può segnare un milione di reti in una partita.

La visualizzazione prodotta da un programma di questo tipo potrebbe essere:

```
USARE 1000000 PER
TERMINARE INPUT
PROSSIMO NUMERO? 5
PROSSIMO NUMERO? 7
PROSSIMO NUMERO? Ø
PROSSIMO NUMERO? 2
PROSSIMO NUMERO? 1
PROSSIMO NUMERO? 1000000
MEDIA 3
BREAK...
```

Risposte battute dall'utente

Per usare questo sistema dobbiamo risistemare l'intero schema di flusso, in particolare la domanda "altri dati" deve venire *prima* del blocco che elabora ciascun elemento – altrimenti il valore che pone termine verrebbe trattato come un elemento normale e comprometterebbe il sommario.

Il corrispondente programma per trovare una media è abbastanza lineare:

```
10 PRINT "USARE 1000000 PER"
20 PRINT "TERMINARE INPUT"
30 S=Ø
40 N=Ø
50 INPUT "PROSSIMO NUMERO"; X
60 IF X = 1000000 THEN 100
70 S=S+X
80 N=N+1
90 GOTO 50
100 PRINT "MEDIA="; S/N
110 STOP
```

Glossario

S: È usato per sommare i valori degli elementi

N: È usato per contare gli elementi

X: È usato per immettere i singoli elementi

Per riassumere, sono stati esaminati quattro diversi modi per indicare quanti elementi d'informazione devono essere acquisiti da un programma. Essi sono:

1. Il numero degli elementi è specificato dal programmatore. È usato solo dai principianti e in pratica è inutile.
2. Il numero degli elementi è specificato in anticipo dall'utente. Un buon metodo se ci sono 20 elementi o meno.
3. L'utente indica dopo ciascun elemento se ce ne sono altri. Intollerabilmente noioso.
4. Il flusso di elementi termina con un valore speciale. Un buon metodo, generalmente migliore degli altri.

ESERCIZIO 14.1

Scrivere un semplice programma che richiede in input il vecchio saldo e gli estremi di tutti gli assegni staccati e quindi visualizza il nuovo saldo o lo scoperto. Usare lo zero come terminatore in quanto non si scriverà mai un assegno di Ø lire. Non tener conto dei crediti. Progettare il programma in modo che sia in grado di produrre una delle due soluzioni che seguono:

(a) VECCHIO SALDO? 550000

BATTERE ESTREMI DEGLI

ASSEGNI, USARE Ø PER TERMINARE

IMPORTO? 100000

IMPORTO? 200000

IMPORTO? Ø

SALDO 250000

Risposte battute dall'utente

(b) VECCHIO SALDO? 500000

BATTERE ESTREMI DEGLI

ASSEGNI, USARE Ø PER TERMINARE

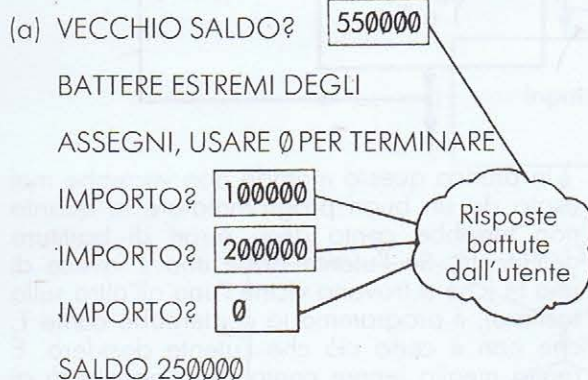
IMPORTO? 400000

IMPORTO? 300000

IMPORTO? 200000

IMPORTO? Ø

SCOPERTO È 400000

Risposte battute dall'utente

Suggerimento: La sezione di programma relativa alla visualizzazione sarà più complessa del solito. Se B è una variabile che dà il saldo corrente, sarà negativa (o minore di Ø) se si preleva più del saldo disponibile. La condizione corretta per controllare questa possibilità è B<Ø. La soluzione dovrebbe comprendere come al solito uno schema di flusso e un glossario. Controllare la risposta sull'Appendice B.

Esercizio 14.1 completato

Talvolta si presenta la necessità di trattare in maniera differenziata i singoli elementi di un flusso di dati. I corrispondenti programmi generalmente hanno dei comandi IF all'interno delle iterazioni principali. Supponiamo ad esempio che, dopo una partita sfortunata si abbia il sospetto che una monetina fosse truccata in modo da dare testa più spesso di croce. Si potrebbe verificare questo sospetto lanciando la monetina un grande numero di volte e contando il numero di teste e croci uscite, servendosi del computer per tenere più facilmente il conteggio dei risultati. Si potrebbe scrivere un programma in grado di dare una visualizzazione di questo tipo:

BATTI H PER TESTA

BATTI T PER CROCE

BATTI E PER TERMINARE

TOTALI: 547

TESTA: 490 VOLTE

CROCE: 57 VOLTE

READY.

In questo modo si potrebbero ricavare le proprie conclusioni sul trucco della monetina.

Progettare e scrivere questo programma a partire dalla stesura del glossario e dello schema di flusso fino ai comandi BASIC.

La visualizzazione campione mostra che si usa un valore speciale E per terminare il flusso di dati. Lo schema di flusso sarà più o meno come quello di pagina 111 e tutto ciò che occorre fare è espandere le nuvolette.

Il programma necessita ovviamente di 3 variabili:

H: Per contare il numero delle volte in cui esce "testa"

T: Per contare il numero delle volte in cui esce "croce"

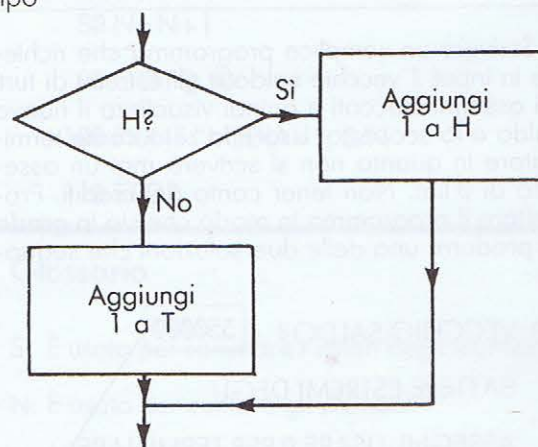
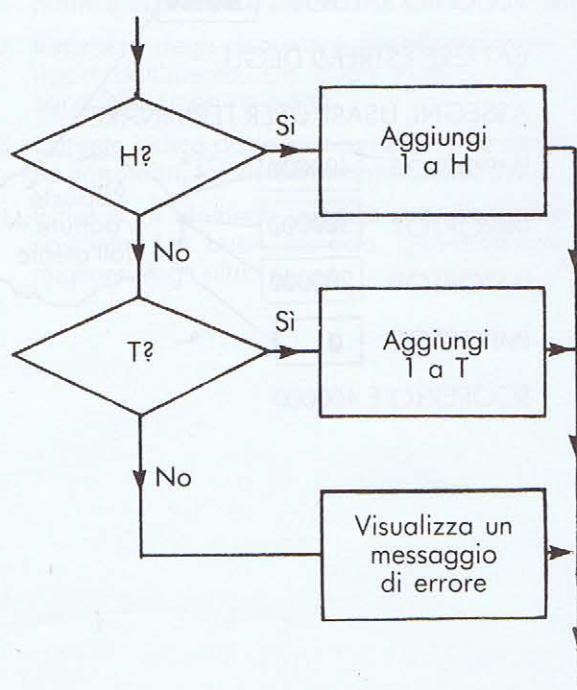
I$: Per immettere un elemento

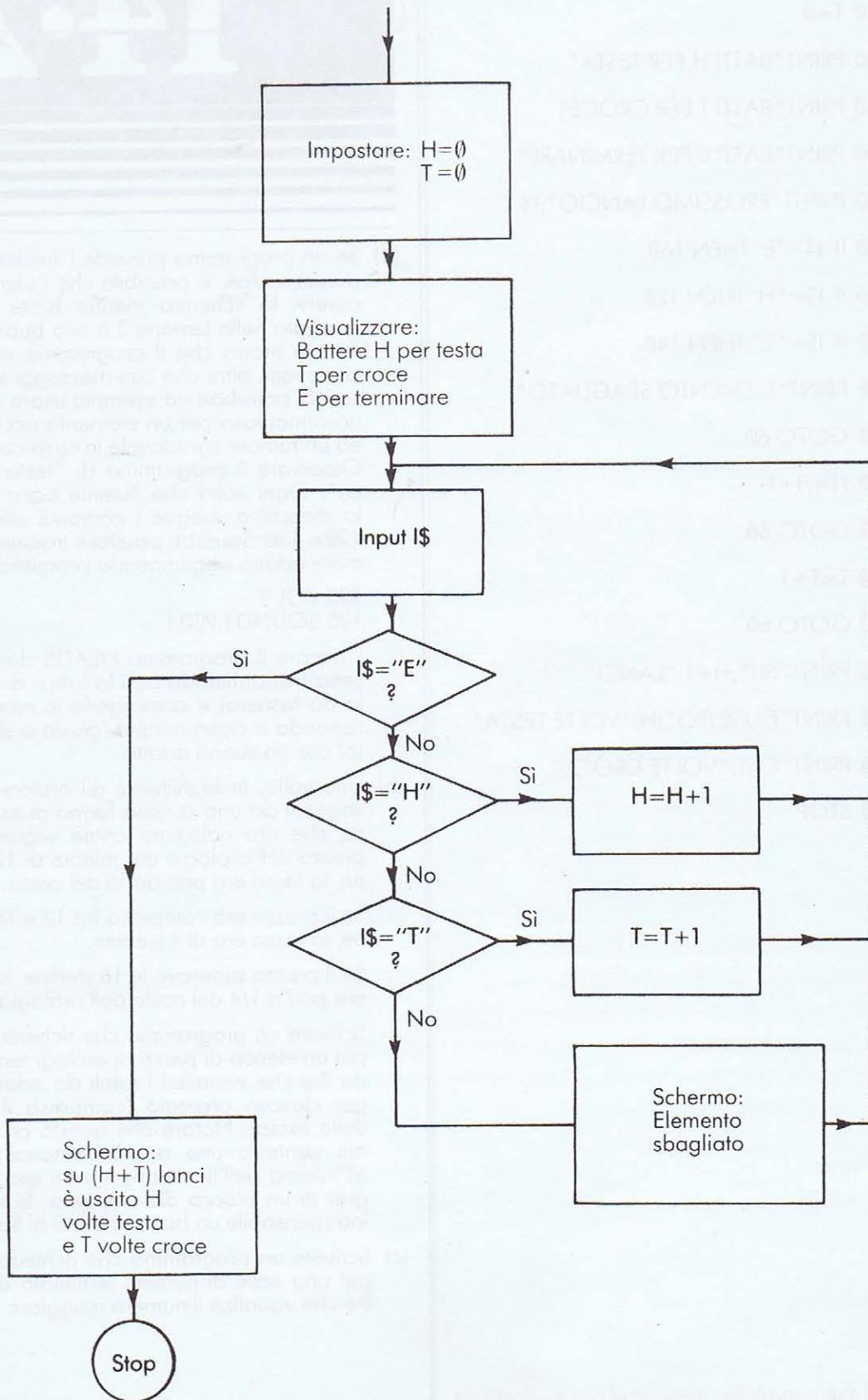
(Come al solito, i nomi H e T sono scelti liberamente).

Alcuni potrebbero essere tentati di prevedere una quarta variabile per contare il numero totale di lanci, ma sarebbe un lavoro inutile; il totale è sempre indicato dall'espressione H+T (numero delle teste + numero delle croci).

A questo punto è possibile elaborare la sezione inizializzazione del programma. Le azioni da eseguire sono due:

- Impostare le variabili H e T a zero
- Visualizzare il messaggio di testa

Si passerà poi alla nuvoletta all'interno dell'iterazione principale che elabora ciascun nuovo elemento. In questa fase la "E" sarà stata filtrata e ogni elemento sarà una H o una T. Il compito fondamentale che la nuvoletta deve svolgere è di aggiungere 1 al totale delle teste o al totale delle croci. Un approccio possibile consisterebbe nell'usare il ragionamento "è un H?, se non lo è deve essere una T". Ciò darebbe luogo ad uno schema di flusso del tipo

In pratica questo metodo non verrebbe mai usato da un buon programmatore in quanto non terrebbe conto degli errori di battitura dell'utente. Se l'utente batte una J invece di una H (che si trovano vicine l'una all'altra sulla tastiera), il programma la conterebbe come T, che non è certo ciò che l'utente desidera. È molto meglio tenere conto della possibilità di errori in questo modo:

Un programma che controlla le conseguenze di eventuali errori dell'utente è detto "solido".

Infine, si può espandere la nuvoletta di sommario per dare un prospetto di tre righe al termine della visualizzazione. Lo schema ampliato apparirebbe come segue:

Qui di seguito è riportato il corrispondente programma. Notare che l'iterazione principale è piuttosto complessa. Ciò è inevitabile dato che occorre forzare uno schema di flusso bidimensionale in una singola serie di istruzioni:

```
10 H=Ø
20 T=Ø
30 PRINT"BATTI H PER TESTA"
40 PRINT"BATTI T PER CROCE"
50 PRINT"BATTI E PER TERMINARE"
60 INPUT"PROSSIMO LANCIO";I$
70 IF I$="E"THEN 16Ø
80 IF I$="H"THEN 12Ø
90 IF I$="T" THEN 14Ø
100 PRINT"ELEMENTO SBAGLIATO"
110 GOTO 6Ø
120 H=H+1
130 GOTO 6Ø
140 T=T+1
150 GOTO 6Ø
160 PRINT"SU";H+T;"LANCI"
170 PRINT"È USCITO";H;"VOLTE TESTA"
180 PRINT"E";T;"VOLTE CROCE"
190 STOP
```


ESERCIZIO 14·2

(a) Se un programma prevede l'immissione di parecchi dati, è possibile che l'utente non osservi lo schermo mentre batte. Come spiegato nella Lezione 3 è una buona idea fare in modo che il programma reagisca con suoni, oltre che con messaggi visualizzati. Si potrebbe ad esempio usare un suono armonioso per un elemento accettabile ed un rumore sgradevole in caso contrario. Osservare il programma di "testa o croce". Ogni volta che l'utente batte una H la macchina esegue i comandi alle righe 12Ø e 13Ø. Sarebbe possibile inserire un rumore adatto aggiungendo i comandi:

```
123 VOL 7
125 SOUND1,9ØØ,1
```

Caricare il programma HEADS dalla cassetta (risparmiando così la fatica di batterlo da tastiera) e correggerlo in modo che risponda a ciascun input (giusto o sbagliato) con un suono adatto.

(b) Una volta, in Inghilterra, gli orologi erano soggetti ad una curiosa forma di tassazione che era calcolata come segue: se il prezzo dell'orologio era minore di 12 sterline, la tassa era pari ad ⅓ del costo.

Se il prezzo era compreso fra 12 e 16 sterline, la tassa era di 4 sterline.

Se il prezzo superava le 16 sterline, la tassa era pari a 1/4 del costo dell'orologio.

Scrivere un programma che richieda in input un elenco di prezzi di orologi terminato da Ø e che visualizzi i totali da addebitare per ciascun orologio (compreso il costo della tassa). Notare che questo programma conterrà uno o più comandi PRINT all'interno dell'iterazione e non avrà bisogno di un blocco di sommario. Si troverà indispensabile un buon schema di flusso.

(c) Scrivere un programma che richieda in input una serie di numeri, terminata da uno Ø e che visualizzi il numero maggiore.

Suggerimento: usare una variabile per registrare il numero maggiore incontrato fino a quel momento e aggiornarla ogni volta che viene eseguita l'iterazione.

Esercizio 14.2 completato	

Controllare ora le risposte nell'Appendice B.

LEZIONE: 15

Questa Lezione tratta tre importanti caratteristiche del BASIC Commodore che sono utili nei giochi, nei quiz e in altri programmi in cui c'è una stretta interazione tra il computer e l'utente.

Si inizierà dando uno sguardo a "REACTION" uno dei programmi che si trovano su cassetta o su dischetto. Il "tempo di reazione" è una misura della rapidità con la quale una persona risponde ad un evento inaspettato. Un guidatore esperto deve avere un breve tempo di reazione in modo da poter azionare rapidamente i freni, ad esempio quando un bambino attraversa la strada davanti alla sua vettura. Un buon tempo di reazione è anche utile nella maggior parte degli sport e in molte professioni.

Molte persone, se fanno attenzione, hanno tempi di reazione fra 0,2 e 0,3 secondi (da 20 a 30 centesimi di secondo). Un tempo di reazione minore di 0,2 fa pensare ad una persona particolarmente sveglia mentre un tempo di più di 0,3 secondi è tipico di chi ha bevuto qualche bicchiere di troppo.

ESERCIZIO 15·1

Caricare il programma REACTION e usarlo per misurare il proprio tempo di reazione. Eseguire il programma più volte e ignorare i primi due o tre risultati, dato che saranno tipici. Continuare a provare il programma fino a che non lo si è compreso fino in fondo e lo si può usare con fiducia per misurare il tempo di reazione di un amico che non ha conoscenze di programmazione.

Vi sono tre aspetti del programma che non sono di comprensione immediata.

Primo: quando viene visualizzato il messaggio "qualsiasi tasto", questo è valido per i tasti funzione come RETURN e INST DEL, e naturalmente anche per i tasti relativi alle lettere e ai numeri, ma non per i tasti C=, SHIFT e CTRL che non avranno alcun effetto.

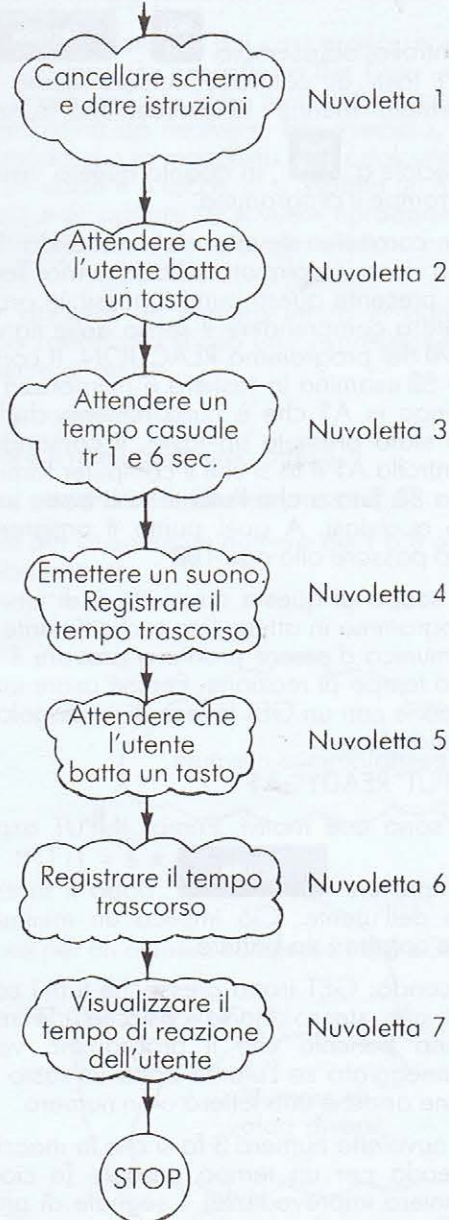
Gli unici tasti da non usare sono:

F1 F4, F2 F5, F3 F6 e HELP F7,

dal momento che, alla loro pressione, vengono prodotti più caratteri (lettere o numeri).

Secondo: il tempo di attesa prima della segnalazione acustica varia in modo imprevedibile da 1 a 6 secondi.

Terzo: se si preme un tasto prima che inizi la segnalazione acustica si ottiene il messaggio "TOO SOON" (troppo presto).

Si esaminerà ora il programma in dettaglio e si spiegherà come funziona. Iniziamo con l'esaminare lo schema di flusso e il programma BASIC, indicati qui di seguito:

Nuvoletta 1

```
10 REM PROGRAMMA TEMPO DI
   REAZIONE
20 PRINT"[SHIFT] e [CLR HOME]"
30 PRINT"BATTERE UN TASTO
   QUALSIASI PER MISURARE
   IL TEMPO DI REAZIONE"
40 PRINT"BATTERE E ATTENDERE
   IL SEGNALE ACUSTICO"
50 PRINT"DOPO IL SEGNALE
   ACUSTICO BATTERE UN
   TASTO QUALSIASI"
60 PRINT"PIÙ IN FRETTA POSSIBILE.
   BUONA FORTUNA!"
```

Nuvoletta 2

```
70 REM ATTESA DI UN TASTO
   QUALSIASI
80 GET A$
90 IF A$=""THEN 8Ø
```

Nuvoletta 3 (e vedere più avanti)

```
100 REM ATTESA DI UN TEMPO
    CASUALE
110 PRINT
120 PRINT "ATTENDERE"
130 PRINT
140 Q=TI+INT(6Ø+3Ø1★RND(Ø))
150 GET A$
160 IF A$<>""THEN 31Ø
170 IF TI<Q THEN 15Ø
```

Nuvoletta 4

```
180 REM ATTIVAZIONE SEGNALE
    ACUSTICO E REGISTRAZIONE
    TEMPO
190 VOL 7
200 SOUND 1,95Ø,4
210 X=TI
```

Nuvoletta 5

```
220 REM ATTESA DI UN
    TASTO QUALSIASI
230 GET A$
240 IF A$=""THEN 23Ø
```

Nuvoletta 6

```
250 REM ACQUISIZIONE RISULTATO
260 R=TI
```

Nuvoletta 7

```
270 REM VISUALIZZAZIONE
    RISULTATO
280 PRINT"TEMPO DI REAZIONE="
290 PRINT(R–X)/6Ø;"SECONDI"
300 STOP
```

Parte della nuvoletta 3

```
310 PRINT"TROPPO PRESTO"
320 STOP

READY.
```

Il programma è stato contrassegnato in modo che siano chiaramente visibili i comandi che corrispondono a ciascuna nuvoletta nello schema di flusso.

La prima nuvoletta (righe da 1Ø a 6Ø) si compone interamente di comandi PRINT ed è abbastanza lineare.

La seconda nuvoletta, righe da 7Ø a 9Ø, fa sì che il programma attenda fino a che l'utente non batte un tasto. La nuvoletta N.2 usa un comando con una nuova parola chiave:

GET A$

Questo comando è analogo al comando INPUT e trasferisce le informazioni dalla tastiera al computer. Tuttavia ci sono alcune importanti differenze:

1. La parola chiave GET deve essere obbligatoriamente seguita esattamente da un solo nome di variabile. Per esempio:

   GET Q

   GET X$ ma — *Non* è BASIC

   GET PR$ — GET Y$,Z$

   (ammesso) — (non ammesso)

2. Il comando GET non attende un intervento dell'utente, ma esamina semplicemente la tastiera in quell'istante e indica quale tasto è stato battuto dall'esecuzione dell'ultimo comando GET o INPUT. Dall'eventuale battitura di un tasto viene ricavato un carattere o un valore numerico; questo viene poi inserito nella variabile citata nel comando GET. Se non è stato di recente battuto alcun tasto, la variabile viene impostata ad una stringa nulla o a Ø. Si tratta di una stringa senza caratteri, normalmente scritta nella forma " " o Ø per i valori numerici. Per illustrare questa regola immaginare di eseguire sul computer il seguente programma iterativo:

   ```
   1Ø GET X$
   2Ø GO TO 1Ø
   ```

   Il computer esegue questa iterazione circa 50 volte in un secondo. Fintantochè l'utente non tocca la tastiera, X$ viene impostato ad una stringa nulla e cioè: " ".

   Supponiamo ora che l'utente prema un tasto – ad esempio quello contrassegnato U. Non appena viene eseguito il comando GET (e cioè entro un 5Øesimo di secondo) X$ sarà impostato alla stringa "U".

   In ogni caso ciò si verifica soltanto una volta per ogni pressione di tasto, la prossima volta che l'iterazione viene eseguita, X$ sarà di nuovo impostata a " " e ciò continuerà fino a che viene rilasciato il tasto U e viene premuto un altro (o lo stesso) tasto. Le sole eccezioni a questa regola sono i cosiddetti tasti di ripetizione, come la barra spaziatrice.

3. Il comando GET considera alcuni tasti di controllo, ad esempio INST DEL, RETURN e i tasti di controllo cursore come tasti normali, mentre riconosce una funzione speciale a RUN STOP, in quanto questo tasto interrompe il programma.

4. Un carattere rilevato dal comando GET, non viene visualizzato sullo schermo. Tenendo presente questi punti è possibile ora iniziare a comprendere il senso delle righe 8Ø e 9Ø del programma REACTION. Il comando 8Ø esamina la tastiera e memorizza una stringa in A$ che è nulla almeno che non sia stato premuto un tasto. Il comando 9Ø controlla A$ e fa sì che il computer torni alla riga 8Ø, fino a che l'utente non batte un tasto qualsiasi. A quel punto il programma può passare alla riga 1ØØ.

Lo scopo di questa nuvoletta è di tenere il programma in attesa fino a che l'utente non comunica d'essere pronto a provare il proprio tempo di reazione. Perché usare un'iterazione con un GET invece di un singolo comando tipo

INPUT"READY";A$?

Ci sono due motivi. Primo: INPUT aspetta sempre un RETURN dopo il messaggio dell'utente. Ciò implica un minimo di due caratteri da battere.

Secondo: GET tratta pressochè tutti i caratteri allo stesso modo cosicchè c'è molto meno pericolo che il programma venga danneggiato se l'utente batte un tasto funzione anzichè una lettera o un numero.

La nuvoletta numero 3 fa sì che la macchina attenda per un tempo casuale (e cioè in maniera imprevedibile) il segnale di pronto dell'utente e il tono. Il tempo di attesa deve essere variabile, in quanto se fosse sempre lo stesso, l'utente imparerebbe quanto deve attendere prima dell'emissione del tono e non si tratterebbe in questo caso di un evento inaspettato.

La nuvoletta usa due funzioni mai incontrate fino ad ora: la funzione RANDOM (generazione di numeri casuali) e il *temporizzatore* interno.

La funzione RANDOM è un modo per far sì che la macchina produca un numero *imprevedibile**. Ogni volta che la macchina elabora l'espressione RND (Ø) dà un valore diverso compreso fra Ø e 1.

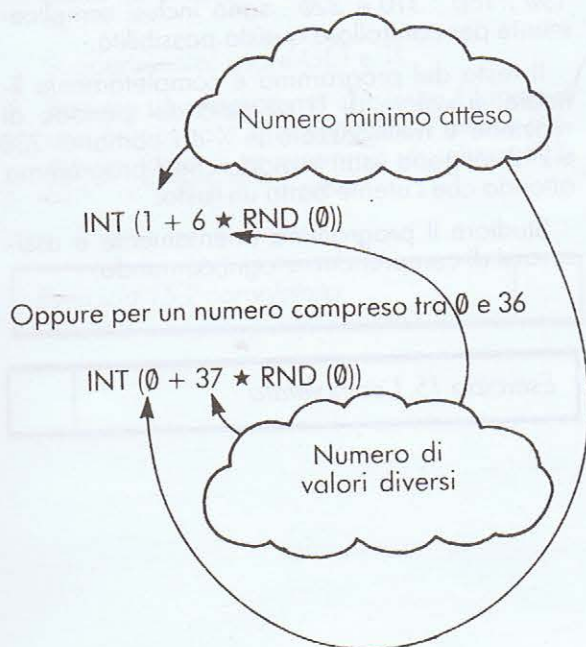
Nella maggioranza dei casi pratici, non occorre una frazione casuale tra Ø e 1 ma un numero intero casuale entro limiti che dipendono dal problema da risolvere. Per esempio, se si fa in modo che la macchina imiti qualcuno che lanci un dado a 6 facce, ci si aspetta un numero tra 1 e 6; oppure se si vuole riprodurre una roulette (europea) occorre un numero compreso tra Ø e 36.

Per ottenere un numero intero in qualsiasi campo specificato si usa una espressione leggermente diversa:

INT(x+y★RND(Ø))

dove x è il numero minimo atteso

y è il *numero delle diverse possibilità*

Così per ottenere un numero da 1 a 6 occorrerebbe inserire

* *Il numero non è realmente imprevedibile in quanto tutto ciò che succede in un computer dipende da ciò che è successo precedentemente. In ogni caso, ciascun nuovo numero "casuale" è derivato dal precedente mediante un processo complicato di elevamento al quadrato e di manipolazione delle cifre del risultato per cui, a meno che non si conosca esattamente l'algoritmo, non si può certamente dire quale sarà il numero che uscirà successivamente.*

Un'espressione di questo tipo può essere inclusa in un'iterazione in modo da poterla eseguire parecchie volte. Battere il seguente programma che imita 12Ø lanci di un dado:

```
NEW

1Ø FOR J = 1 TO 12Ø

2Ø S = INT (1 + 6 ★ RND (Ø))

3Ø PRINT S;

4Ø NEXT J

5Ø STOP
```

Lanciare questo programma e contare il numero degli 1, dei 2... dei 6 che compaiono sullo schermo. Immettere i risultati nella prima fila della tabella che segue:

	1	2	3	4	5	6
Numero lanci (1)						
Numero lanci (2)						

Il programma è una buona imitazione di un gioco di dadi onesto (non truccato)?

Eseguire il programma di nuovo e compilare la seconda fila. Esaminare i risultati e notare che essi differiscono da quelli riportati nella prima fila come ci si aspetterebbe dal gioco reale.

L'altra caratteristica importante della nuvoletta 3 è il temporizzatore interno, TI. Si è già incontrato l'orologio TI$, che conta il tempo in ore, minuti e secondi, ma la variabile speciale TI (che non è una stringa ma un numero) si propone di misurare periodi di tempo molto più brevi. TI è impostato a Ø quando il computer viene avviato e da quel momento in poi, qualunque cosa succeda, viene aggiunto un 1 ogni 60esimo di secondo. Questo intervallo, 1/60 di secondo, è detto un "jiffy". È possibile ottenere il valore corrente in jiffy del temporizzatore interno in qualsiasi momento usando il nome TI in un'espressione, ma non è possibile modificare il suo valore come con TI$.

Impartire il comando

PRINT TI

Il computer risponderà visualizzando un numero abbastanza grande (6Ø★6Ø o 36ØØ jiffy per ogni minuto durante il quale la macchina è stata accesa). Ora provare di nuovo il comando: si noterà che il valore è aumentato di qualche centinaia. Infine provare a ripristinare il valore di TI e osservare il risultato.

TI può essere usato per misurare periodi di tempo in due modi diversi ma correlati, in nessuno dei quali è importante il numero di jiffy trascorsi da quando il computer è stato acceso. È invece rilevante il fatto che la durata di

qualsiasi intervallo di tempo è indicata dalla *differenza* tra i valori di TI all'inizio e alla fine del periodo in questione. Per esempio, al termine di un periodo di 5 secondi, TI sarà maggiore di 5★60 (cioè 300) rispetto al valore che aveva all'inizio del periodo considerato. Ciò è vero sia che la macchina sia stata accesa per 5 secondi o per 5 ore.

Nel primo metodo di uso del temporizzatore interno, si fa in modo che la macchina misuri un periodo di tempo che è noto in anticipo e il computer comunica quanto tempo è trascorso. Il metodo è semplice. All'inizio del periodo il programma considera TI e prevede che valore avrà al termine del periodo; quindi attende in un'iterazione fino a che TI raggiunge (o supera) quel valore. È più o meno quello che si fa in cucina quando si dice "queste patate devono bollire per 25 minuti. Ora sono le 4 e 10 e dovremmo toglierle dal fuoco alle 4 e 35". Questo concetto viene illustrato dal seguente programma temporizzatore per impiego generico che si potrebbe usare in cucina, in laboratorio, ecc.

```
10 INPUT "QUANTI MINUTI": M
20 R=TI+M★3600
30 IF TI < R THEN 30
40 PRINT "TEMPO SCADUTO!"
50 STOP
```

Se si esegue questo programma è consigliabile utilizzare un numero di minuti non troppo grande, per evitare perdite di tempo. Quando si studia il programma occorre ricordare che TI aumenta costantemente fino a che, dopo M★3600 jiffy, la condizione TI<R risulta falsa.

Nella seconda variante si vuole che il computer visualizzi quanto tempo trascorre da un dato momento fino al verificarsi di qualche evento. Si fa in modo che la macchina registri il valore di TI e il valore registrato è una misura dell'intervallo espressa in jiffy. Questa seconda variante ricorda il ragionamento di uno scalatore che pensa: "Ho iniziato a scalare questa collina alle 5 di questa mattina. Ho raggiunto la cima alle 11 e mi sono occorse perciò 6 ore".

Un programma che misurasse il tempo in questo modo avrebbe comandi di questo genere:

R=TI	(Memorizza il valore di TI all'inizio del periodo)
	e successivamente
E=TI	Ottiene il valore di TI al termine del periodo
D=E-R	Ricava la differenza di tempo (in jiffy)
S=D/60	Ricava la differenza di tempo (in secondi)

```
PRINT"SONO STATI IMPIEGATI";S;
    "SECONDI"
```

Ora è possibile riunire i comandi nella nuvoletta numero 3.

Si vuole attendere un periodo compreso tra 1 e 6 secondi. Il che significa tra 60 e 360 jiffy: il valore esatto verrà determinato dal computer in maniera imprevedibile. L'espressione appropriata è:

INT (60+301★RND (0))

Il periodo di attesa è deciso immediatamente prima dell'inizio del periodo stesso, cosicchè è noto in anticipo (ma non all'utente). Si usa il primo metodo di calcolo del tempo che comporta la previsione del valore di TI al termine del periodo. Il comando 140 effettua questa previsione e registra il valore in Q.

Se questo fosse tutto ciò che occorre, l'intera nuvoletta potrebbe comparire come segue:

```
140 Q=TI+INT(60+301★RND(0))
170 IF TI <Q THEN 170
```

In realtà, occorre controllare che l'utente non batta un tasto prima di udire il tono. I comandi 150 , 160 , 310 e 320 , sono inclusi semplicemente per controllare questa possibilità.

Il resto del programma è completamente lineare. Il valore di TI all'inizio del periodo di reazione è memorizzato in X e i comandi 230 e 240 vengono usati in modo che il programma attenda che l'utente batta un tasto.

Studiare il programma attentamente e assicurarsi di comprenderne ogni comando.

Esercizio 15.1 completato	

ESERCIZIO 15.2

1. Scrivere un programma "cronometro". Quando l'utente batte il tasto B, il programma inizia a contare. Quando batte S, si interrompe e visualizza il tempo trascorso in secondi.

   Il programma dovrebbe visualizzare le opportune indicazioni in modo da poter essere usato da chiunque senza ulteriori spiegazioni.

   Suggerimento: usare GET e TI.

2. Scrivere un programma che imita il lancio di una monetina. Ogni volta che l'utente preme un tasto, il programma visualizza "TESTA" o "CROCE" a caso.

Esercizio 15.2 completato	

Controllare ora le risposte nell'Appendice B.

ESERCIZIO 15.3

I numeri casuali sono utili nella programmazione dei giochi di probabilità, ad esempio il lancio dei dadi, il funzionamento di slot machine e così via. Tutti questi programmi seguono lo stesso profilo base per un "lancio" o "colpo":

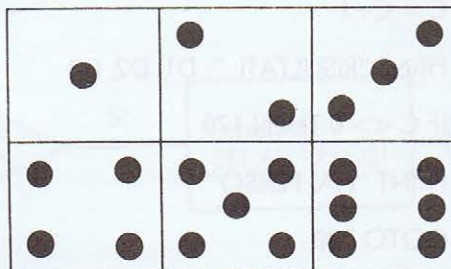
Chiede all'utente i dati relativi alla sua scommessa

↓

Usa numeri casuali per imitare il lancio, colpo, ecc.

↓

Decide se il giocatore ha vinto o perso e glielo dice

Si illustrerà questo concetto con il vecchio gioco della corona e áncora*, che si gioca con tre dadi e una tavoletta divisa in sei quadrati*:

* *Questo gioco usa solitamente diversi simboli ma ciò non influisce sul principio del gioco stesso.*

Il giocatore fa la sua puntata su uno qualsiasi dei quadrati. Per esempio, potrebbe scommettere 5 mila lire su ⚁. Quindi chi tiene il banco getta i tre dadi. Se uno di essi esce con ⚁ il giocatore vince il doppio della sua posta; se due dadi escono con ⚁ il giocatore riceve il triplo della posta originale e se il ⚁ esce su tutti e tre i dadi, il giocatore è remunerato con quattro volte la sua posta. Tutte queste vincite comprendono la posta originale.

Per contro, se non esce alcun ⚁ il giocatore perde la posta. Qui di seguito è indicato il programma per giocare una mano di corona e áncora. Usando il glossario, non si dovrebbero avere difficoltà nel seguirlo.

S:	Posta del giocatore
N:	Numero puntato dal giocatore
D1, D2, D3	Risultato del lancio di 3 dadi
C:	Numero di dadi usciti con N, il numero del giocatore

```
10 INPUT "POSTA"; S
20 INPUT "NUMERO PUNTATO (1-6)"; N
30 D1 = INT (1+6*RND(0))      } Lancio di 3 dadi
40 D2 = INT (1+6*RND(0))
50 D3 = INT (1+6*RND(0))
60 C = 0                      } Conta il numero di dadi
70 IF D1 <> N THEN 90           usciti col numero
80 C = C+1                      puntato dal giocatore
90 IF D2 <> N THEN 110
100 C = C+1
110 IF D3 <> N THEN 130
120 C = C+1
130 PRINT "RISULTATI :"; D1; D2; D3   } Visualizza i risultati
140 IF C <> 0 THEN 170
150 PRINT "HAI PERSO"
160 GOTO 180
170 PRINT "VINCI"; S*(C+1); "LIRE"
180 STOP
```

Pochissimi giocatori interrompono il gioco dopo un lancio. Solitamente si inizia con un certo capitale e si continua a giocare fino a che non si vada in rovina o – molto raramente – finchè salta il banco.

I programmi che simulano giochi d'azzardo sul computer sono ovviamente più verosimili se imitano sessioni complete di questo tipo. Inizialmente al giocatore viene assegnata una certa somma di denaro (es. 100.000 lire) che egli può utilizzare finchè lo desidera o finchè la somma non è completamente esaurita. Qui di seguito è indicato uno schema di flusso per tale tipo di gioco:

Il giocatore riceve il capitale di partenza

Visualizza l'importo del capitale rimanente

Chiede al giocatore la sua puntata

Può permetterselo?

No

Schermo: "NON PUOI SCOMMETTERE PIÙ DI QUELLO CHE TI RIMANE"

Sì

Imita un lancio/colpo

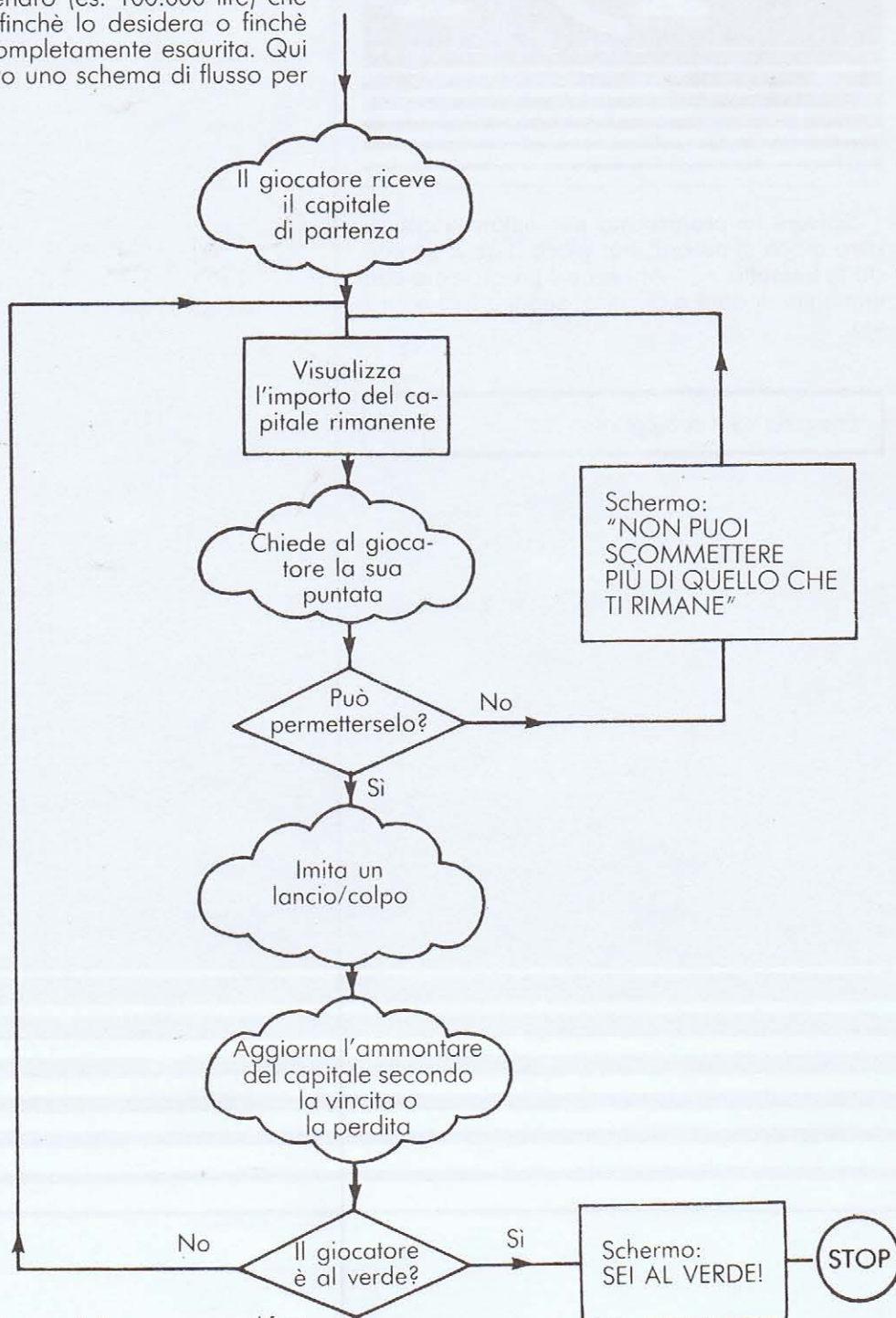
Aggiorna l'ammontare del capitale secondo la vincita o la perdita

Il giocatore è al verde?

No

Sì

Schermo: SEI AL VERDE!

STOP

Usare questo schema di flusso per modificare il programma corona e áncora in modo da far partire l'utente con un capitale di 100.000 lire e consentirgli di giocare finchè lo desidera. Quando il programma è completo, eseguirlo più volte e decidere se è meglio assumere il ruolo del giocatore o quello del banco.

Esercizio 15.3 completato	

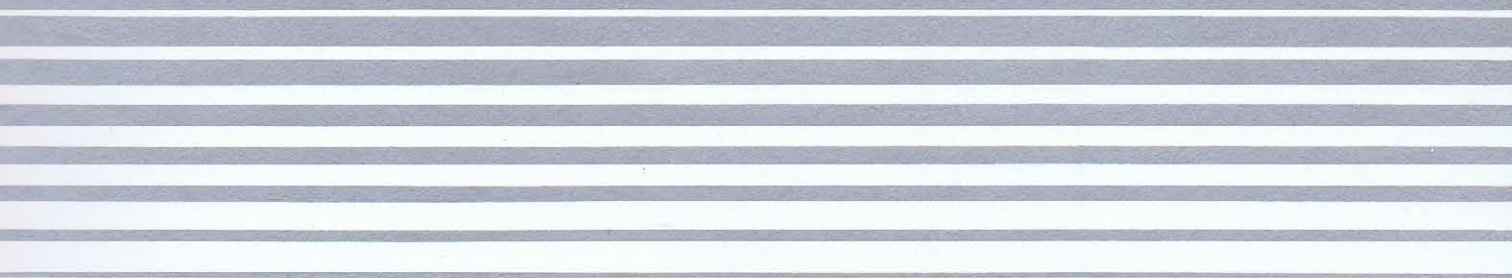
ESERCIZIO 15.4

Scrivere un programma per imitare qualsiasi altro gioco di probabilità: gioco d'azzardo con dadi, tressette, ecc. Abbellire il programma con immagini di dadi o di carte, suoni adatti e così via.

Esercizio 15.4 completato	

Il nastro o il dischetto contengono un programma "dadi" da provare.

Un listato completo del programma, denominato "CRAPS" (dadi) viene dato nell'Appendice B.

CONCLUSIONE

Congratulazioni per aver raggiunto la fine del corso. A questo punto avrete certamente acquisito una buona conoscenza dei principi della programmazione e sarete in grado di progettare e scrivere programmi per una vasta gamma di problemi e di applicazioni interessanti. Ci auguriamo che abbiate anche preso l'abitudine ad una progettazione accurata e meditata, alla conservazione e all'archiviazione degli schemi di flusso, dei glossari e delle note per i programmi.
Sono queste le qualità di pianificazione e di auto-organizzazione che distinguono il programmatore veramente competente dagli altri.

A questo punto avete compiuto metà della strada nello studio del BASIC. Ci sono molti altri problemi che richiedono parti del linguaggi che non sono state ancora trattate. Per esempio, si possono far muovere immagini sullo schermo o riordinare nomi di persone in ordine alfabetico oppure memorizzarli sulla cassetta o sul floppy disk. Questi argomenti e molti altri sono completamente spiegati nel secondo libro di questa serie intitolato

INTRODUZIONE AL BASIC (Parte II)

concepito secondo gli stessi criteri di quello che avete appena terminato e che completerà la vostra conoscenza del linguaggio BASIC.

La programmazione – come è stato detto nell'introduzione – è un campo molto vasto che dopo una buona preparazione di base può essere approfondito in tre modi:

(a) Leggendo il più possibile. Vale la pena di leggere molte riviste sull'argomento, specialmente Commodore Computer Club o libri sulla programmazione, anche se questi spesso non fanno riferimento ad una specifica macchina.

(b) Associandosi ad un club locale di appassionati di computer. Troverete gli indirizzi su Computer Club.

(c) Esercitandosi praticamente nella programmazione senza accontentarsi mai dei risultati ottenuti. È opportuno progettare i propri programmi cercando di renderli "solidi" e utilizzabili da chiunque senza speciali indicazioni.
È opportuno inoltre scrivere i programmi in modo che la loro stesura sia di facile comprensione anche per un altro programmatore.

Un ultimo punto. Avete senz'altro trovato un hobby affascinante e forse anche una professione per il futuro. Ricordate che il fatto di essere un esperto di computer comporta, assieme ai vantaggi, anche delle responsabilità in quanto è auspicabile che le macchine vengano usate in maniera umana e possibilmente anche saggia. Nessuno vorrebbe certamente una società controllata dal computer con poco lavoro ma senza libertà. Tocca a voi – fra gli altri – evitarlo.

APPENDICI

I computer Commodore **16** e **PLUS/4** sono forniti di ampie possibilità matematiche; per avere un riscontro con i modelli del passato, si può dire che possono eseguire operazioni aritmetiche con una velocità considerevolmente più elevata di gran parte dei computer di grosse dimensioni installati prima del 196Ø.

Questa appendice illustra alcune delle funzioni matematiche del computer. Se si intende utilizzare il computer per calcoli matematici, scientifici e tecnici è sufficiente leggere e comprendere il contenuto dell'appendice. Alcune delle funzioni descritte sono abbastanza semplici e possono essere facilmente comprese da chiunque abbia qualche nozione di aritmetica. Altre funzioni richiedono conoscenze di base più approfondite, equivalenti a quelle fornite nella maggior parte delle scuole medie superiori. Ad ogni modo, procedere nello studio finché le proprie conoscenze lo consentono, posto, comunque, che siano già state lette tutte le Lezioni che costituiscono questo corso.

1. Espressioni

Le espressioni menzionate per la prima volta nella Lezione 4 sono esempi molto semplici di una funzione più generale. Così nei comandi

A=<u>34</u>

B=<u>B+1</u>

C=<u>((X+Y) – 34.7/ (Q–3))★(Z–3) ↑ 2</u>

le parti sottolineate sono tutte espressioni che il computer calcola per conto dell'utente.

Le espressioni sono costituite da tre tipi di elementi:

Valori: variabili numeriche o numeri come

B,X,Y,34,34.7

Operatori: i segni + – ★/ e ↑
(↑ significa "elevato a potenza")

Parentesi: (e)

Le espressioni in BASIC sono scritte come in algebra e hanno lo stesso significato. Ci sono, però, quattro piccole differenze:

- Le espressioni in BASIC sono in lettere maiuscole anziché minuscole.
- L'elevamento a potenza deve essere visualizzato con il segno ↑, poiché lo schermo del computer non consente di scrivere numeri più piccoli al di sopra della riga. Perciò, invece di "3^2", occorre battere "3 ↑ 2".
- La moltiplicazione viene eseguita visualizzando il segno ★. Nel BASIC si scrive, ad esempio, "3★A" e non "3A" come nell'algebra convenzionale.
  Se si ignora questa regola fondamentale, si può incorrere in numerosi errori. Se si batte BA invece di B★A, il computer interpreterà ciò che è stato battuto come una nuova variabile, chiamata BA. Perciò, non visualizzerà un messaggio di errore di sintassi, ma ovviamente non fornirà la risposta dovuta.
- La divisione viene scritta A/B e non $\frac{A}{B}$.

  Se il numeratore o il denominatore della frazione è costituito da un'espressione complessa, questa va posta fra parentesi. Il modo corretto di scrivere $\frac{3+5}{7+8}$ in BASIC è (3+5)/(7+8). Se si tralasciano le parentesi e si pongono le espressioni nella forma 3+5/7+8 le regole della precedenza (che vengono fornite nel prossimo paragrafo) faranno in modo che il computer consideri l'espressione come $3+\frac{5}{7}+8$.

Quando il computer esegue il calcolo di un'espressione, considera per primo il segno ↑, quindi le moltiplicazioni e le divisioni ed infine le addizioni e le sottrazioni, muovendosi in ognuno dei casi da sinistra a destra. Tutto ciò che si trova fra parentesi viene calcolato per primo. Queste vengono definite le regole della precedenza e sono le stesse che vengono usate in algebra.

Il valore dei numeri delle espressioni non deve necessariamente essere intero, ma può anche essere decimale. Il computer esegue i suoi calcoli con una precisione di circa 8 cifre decimali, il che significa che molte frazioni (come, ad esempio, 1/3 o 1/7) non possono essere rappresentate con esattezza. Ci si possono aspettare anche dei piccoli errori di arrotondamento in alcuni comandi aritmetici, così che un risultato che ci si attendeva fosse 7, può essere invece "6.99999998".

Per verificare di aver ben compreso le norme relative alle espressioni, calcolare le espressioni contenute nell'esempio seguente, cercando di prevedere ciò che il computer visualizzerà di volta in volta. Supponiamo che X=3 e P=7.

COMANDO	RISULTATO PREVISTO	RISULTATO EFFETTIVO
PRINT 3 + 12 — 6 — 4		
PRINT 4 + 3 ★ 2		
PRINT X + P — 3		
PRINT 5 + 12 / 6 — 3		
PRINT 11 / 5 — 7 / 4		
PRINT 4 ↑ 2 — 2 ↑ 4		
PRINT 3 + 2 ↑ 3 — 3 ↑ 2		
PRINT 2 ↑ X — P		
PRINT 3 + 12 — (6 — 4)		
PRINT 5 + 12 / (6 — 3)		
PRINT (P + X) ↑ (1 — X)		
PRINT 4 ↑ 2 — 3 ↑ Ø		
PRINT (P ↑ 2 — X ↑ 2) / 3		

Verificare ora i risultati sul computer. Ricordare di impostare i valori di P e di X prima di cominciare.

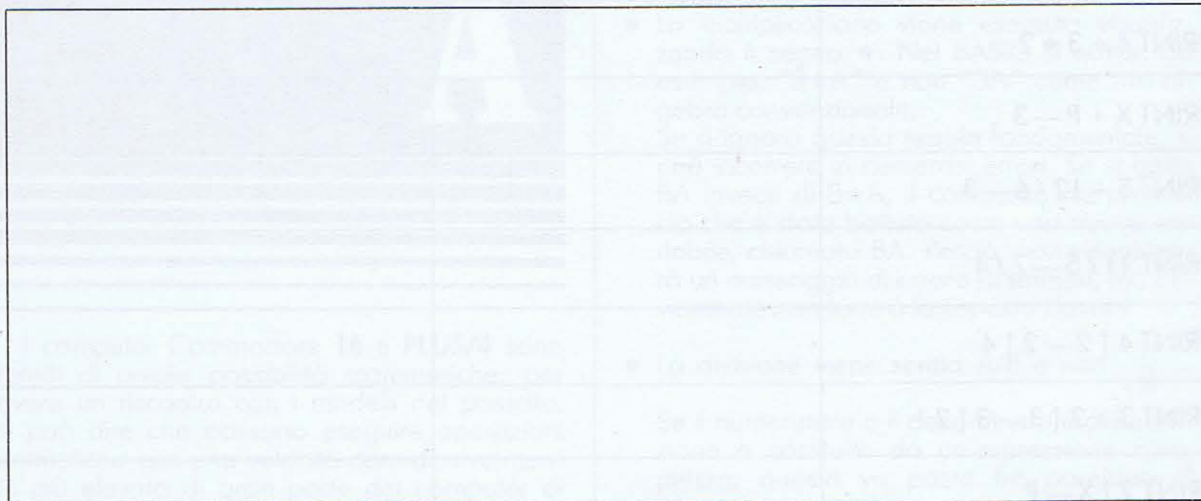
Nel BASIC, le espressioni vengono comunemente usate nei comandi PRINT e LET. Qui di seguito viene fornito un semplice programma che introduce due numeri U e V e visualizza un valore F calcolato secondo la formula della lente

$$\frac{1}{F} = \frac{1}{V} + \frac{1}{U}, \text{ o } F = \frac{1}{\frac{1}{V} + \frac{1}{U}}$$

```
1Ø INPUT "V";V
2Ø INPUT "U";U
3Ø PRINT "F=";1(1/V+1/U)
4Ø STOP
```

Esempio 1

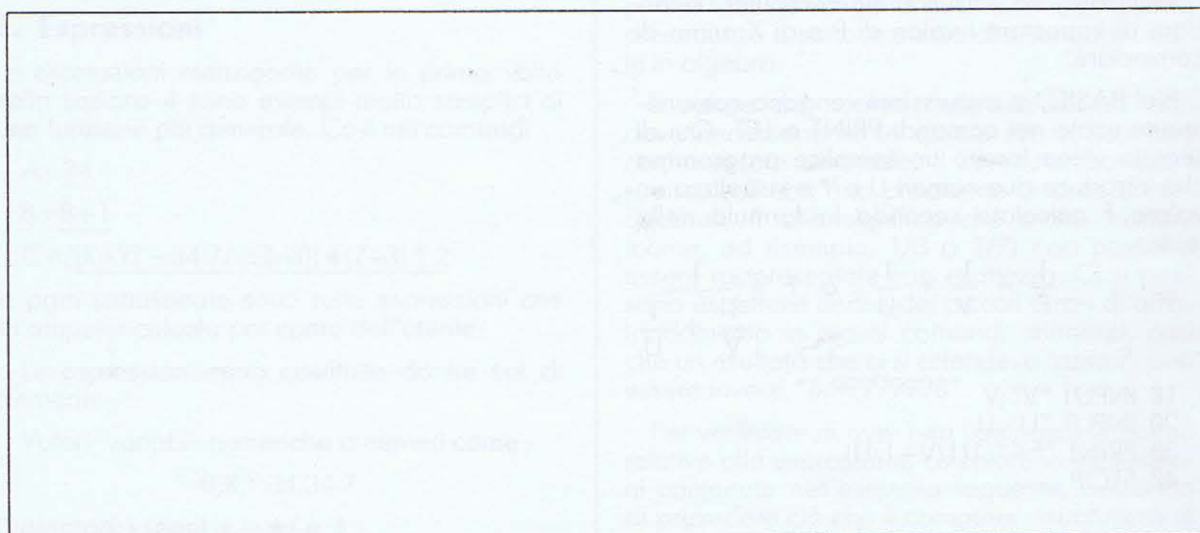
Scrivere un programma che legge i due valori V e R e che visualizza il valore della formula

$$A= \frac{V^2}{R}$$

Esempio 2

Scrivere un programma che visualizza i valori

della formula $Y= \frac{1}{1 + x^2}$ per tutti i valori di X che sono compresi fra Ø e 2, e ciascuno dei quali è uguale al precedente +0.2.

(Si suggerisce di usare un ciclo "FOR...NEXT" come quello che segue:

```
FOR X=Ø TO 2 STEP Ø.2
...
NEXT X)
```

(Le risposte corrette vengono fornite in fondo all'Appendice B.)

2. Funzioni Standard

Come la maggior parte delle calcolatrici, il computer è corredato di una serie di funzioni "scientifiche", ad esempio la radice quadrata, che viene abbreviata in SQR e può essere inclusa in espressioni del tipo:

```
PRINT SQR (5)
```

o

```
PRINT SQR (B ↑ 2+C ↑ 2)
```

La quantità fra parentesi viene chiamata argomento della funzione. Nel caso della funzione SQR l'argomento deve essere lo zero o un numero positivo.

Qui di seguito viene fornito un programma che visualizza le radici quadrate di tutti i numeri compresi fra 1ØØ e 115.

```
1Ø PRINT "N";"SQR(N)"
2Ø FOR N=1ØØ TO 115
3Ø PRINT N; SQR(N)
4Ø NEXT N
5Ø STOP
```

Esempio 3

Se le lunghezze dei lati di un triangolo sono a,b e c, l'area A del triangolo è data dalla formula $A=\sqrt{s(s-a)\ (s-b)\ (s-c)}$ dove s è il semiperimetro, (a+b+c)/2.

Scrivere un programma che introduce tre valori. Se questi valori corrispondono alle lunghezze dei lati di un triangolo possibile, il programma visualizza l'area del triangolo; in caso contrario (ad esempio se in numeri sono 1,1,1Ø) il programma visualizza un messaggio appropriato. Se le tre linee impostate non formano un triangolo, il valore dell'espressione s(s–a) (s–b) (s–c) è negativo.

Qui di seguito vengono fornite alcune delle più importanti funzioni matematiche. Si consiglia di leggerle attentamente, ma non è necessario impararle a memoria, dal momento che si può sempre consultare questa lista.

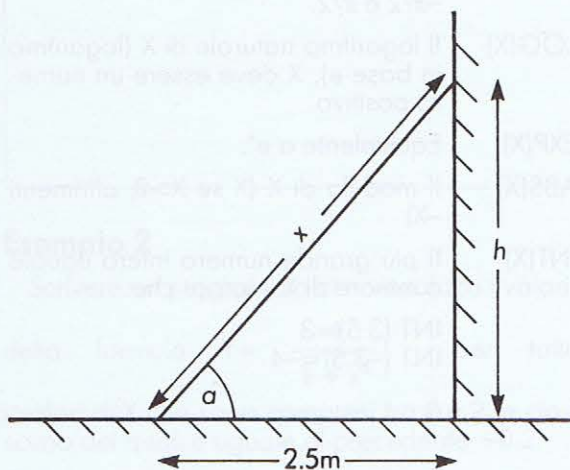
SIN(X) COS(X) TAN(X)	Funzioni trigonometriche. L'argomento deve essere espresso in radianti (1 grado= π/18Ø radianti).
ATN(X)	L'arcontangente di X. Il risultato è espresso in radianti, compresi fra $-\pi/2$ e $\pi/2$.
LOG(X)	Il logaritmo naturale di X (logaritmo in base e). X deve essere un numero positivo.
EXP(X)	Equivalente a e^x.
ABS(X)	Il modulo di X (X se X>Ø; altrimenti –X)
INT(X)	Il più grande numero intero uguale o minore di X. Notare che: INT (3.5)=3 INT (–3.5)=–4

Verificare ora la propria risposta con quella fornita nell'Appendice B.

Al posto del numero 3.14149265...., si può utilizzare il simbolo π.

Di seguito viene fornito un esempio che illustra l'uso di alcune di queste funzioni.

Una scala può avere una lunghezza variabile da 4 a 5 metri, con intervalli di 2Ø cm. Viene collocata con la base a 2,5 m da un muro verticale e con la sommità contro il muro stesso. Scrivere un programma che visualizzi l'angolo formato dalla scala con il piano orizzontale per ognuna delle possibili 6 lunghezze.

Per prima cosa si svolge la parte matematica, aiutandosi con l'uso di un diagramma. Si usa X per indicare la lunghezza della scala, H la distanza dal suolo della sommità della scala stessa e A l'angolo formato con la superficie orizzontale.

$h= \sqrt{X^2-(2.5)^2}$ (secondo il teorema di Pitagora)

a=arc tan (h/2.5) (in radianti)

o

a=(18Ø/π) ★arc tan (h/2.5) in gradi.

Quindi scrivere il programma, che ha una semplice struttura ciclica:

```
10 PRINT"LUNGHEZZA,"ANGOLO"
20 FOR X=4 TO 5 STEP Ø.2
30 H=SQR (X↑2–2.5↑2)
40 A=(18Ø/π)★ATN (H/2.5)
50 PRINT X,A
60 NEXT X
70 STOP
```

Una delle funzioni più utili è INT, che può essere utilizzata per verificare se un numero è divisibile esattamente per un altro. Se X è un multiplo esatto di Y, allora la condizione

X/Y=INT (X/Y)

sarà vera; in caso contrario, sarà falsa.

Un numero è detto numero primo se non ha altri divisori all'infuori di se stesso e di 1. Il programma seguente calcola e visualizza tutti i numeri primi compresi fra 3 e qualsiasi valore impostato dall'utente:

```
10 INPUT"VALORE MASSIMO";H
20 FOR N=3 TO H
30 FOR J=2 TO N–1
40 IF N/J=INT (N/J) THEN 70
50 NEXT J
60 PRINT N;
70 NEXT N
80 STOP
```

Esempio 4

Studiare il programma dei numeri primi (disegnandone eventualmente un diagramma) ed impararne il funzionamento. Eseguirlo, assegnando ad U un certo valore, ad esempio 5ØØ e controllare il tempo impiegato.

In realtà, questo metodo di calcolare i numeri primi non è molto rapido. Progettare ed aggiungere dei miglioramenti atti ad accelerarne l'esecuzione.

Ricordare che:

(a) Nessun numero pari, tranne 2, può essere un numero primo.

(b) Quando si controllano i vari fattori possibili è sufficiente arrivare alla radice quadrata del numero in questione.

Verificare la propria risposta con quella fornita nell'Appendice B.

APPENDICE B

LEZIONE: 7

Esercizio 7.1:

a) T,T,T,T,F,F,F
b) F,F,T,T

Esercizio 7.3:

(1)
```
10 P$="★"
20 PRINT P$
30 P$=P$+"★"
40 IF P$<>"★★★★★★★★★★★"
   THEN 20
50 STOP
```

(2)
```
10 PRINT"STERLINE","DOLLARI"
20 PRINT
30 P=10
40 PRINT P,1.77★P
50 P=P+2
60 IF P<32 THEN 40
70 STOP
```

(3)
```
10 PRINT"CENT","FAHR"
20 PRINT
30 C=15
40 F=1.8★C+32
50 PRINT C,F
60 C=C+1
70 IF C<31 THEN 40
80 STOP
```

Esercizio 7.2:

Variabile di controllo	*Valore iniziale*	*Valore finale*	*Incremento*	*N. di "loop" eseguiti*
X$	"A"	"ABBB"	"B"	4
P	Ø	1Ø	+1	11
Y$	"Z"	"ZXYXY"	"XY"	3
R	5	14	3	4
C	27	7	—5	5

LEZIONE:8

Esercizio 8.1:

a) CONTATORE DI PROGRAMMA ~~10 20 30 40 50~~ 6Ø

VARIABILI X: 5 Y: 7 Z: 12 W: 2

5 7 12 2 BREAK IN 6Ø READY	1Ø X=5 2Ø Y=7 3Ø Z=X+Y 4Ø W=Y−X 5Ø PRINT X; Y; Z; W 6Ø STOP

CONTATORE DI PROGRAMMA ~~10 20 30 40 50 30 40 50~~ 6Ø

VARIABILI Q: ~~1 2~~ 3

LEI MI AMA LEI NON MI AMA LEI NON MI AMA BREAK IN 6Ø READY	1Ø Q=1 2Ø PRINT"LEI MI AMA" 3Ø PRINT"LEI NON MI AMA" 4Ø Q=Q+1 5Ø IF Q < 3 THEN 3Ø 6Ø STOP

Esercizio 8.2:

a) La linea 5Ø dovrebbe essere:
   5Ø IF G<11 THEN 3Ø

b) La linea 3Ø dovrebbe essere:
   3Ø A$=A$+"★"

Esercizio 8.3:

c) Linea 2Ø:PRINT (non PR1NT)
   Nessun RETURN dopo la linea 4Ø
   Linea 6Ø: IF X<13 THEN 4Ø (NON X>13)
   Linea 7Ø: STOP (non STØP)

LEZIONE:9

Esercizio 9.2:

```
10 COLOR4,5
20 COLOR0,8
30 PRINT" SHIFT e CLR HOME ";
40 PRINT" CLR HOME ↓ ← 9 volte → ↓
→ ← 6 volte → → ";
50 PRINT" ← "TI$
60 GOTO 40
```

Esercizio 9.3:

```
 5 REM BANDIERA ISLANDESE
10 COLOR4,1
20 COLOR0,15
30 PRINT" SHIFT e CLR HOME ";
40 J=1
50 PRINT" CTRL e RVS ON
→ ← 6 volte → → CTRL e
WHT ← 1 spazio → CTRL e RED
← 2 spazi → CTRL e 1 spazio"
60 J=J+1
70 IF J<10 THEN 50
80 PRINT" CTRL e RVS ON CTRL e
WHT ←7 spazi→ CTRL e RED
← 2 spazi → CTRL e WHT
← 13 spazi →";
90 J=1
100 PRINT" CTRL e RVS ON CTRL e
RED ← 22 spazi→";
110 J=J+1
120 IF J<4 THEN 100
130 PRINT" CTRL e RVS ON CTRL e
WHT ← 7 spazi → CTRL e RED
← 2 spazi → CTRL e WHT
← 13 spazi →";
140 J=1
150 PRINT" CTRL e RVS ON
→ ← 6 volte → →
CTRL e WHT ← 1 spazio →
CTRL e RED ← 2 spazi →
CTRL e WHT ← 1 spazio →"
160 J=J+1
170 IF J<9 THEN 150
180 PRINT" CTRL e RVS ON
→ ← 6 volte → →
CTRL e WHT ← 1 spazio →
CTRL e RED ← 2 spazi →
CTRL e WHT ← 1 spazio →";
190 GOTO 190
```

LEZIONE:10

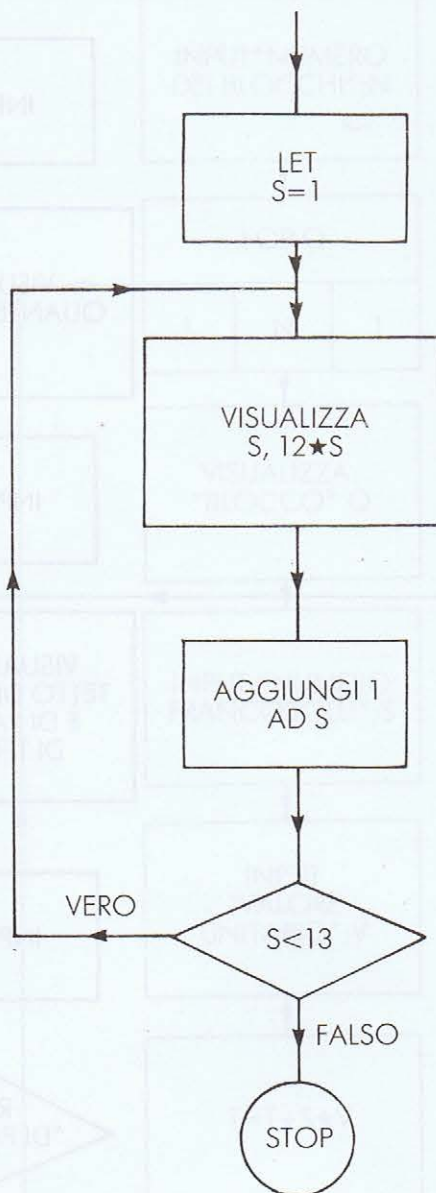
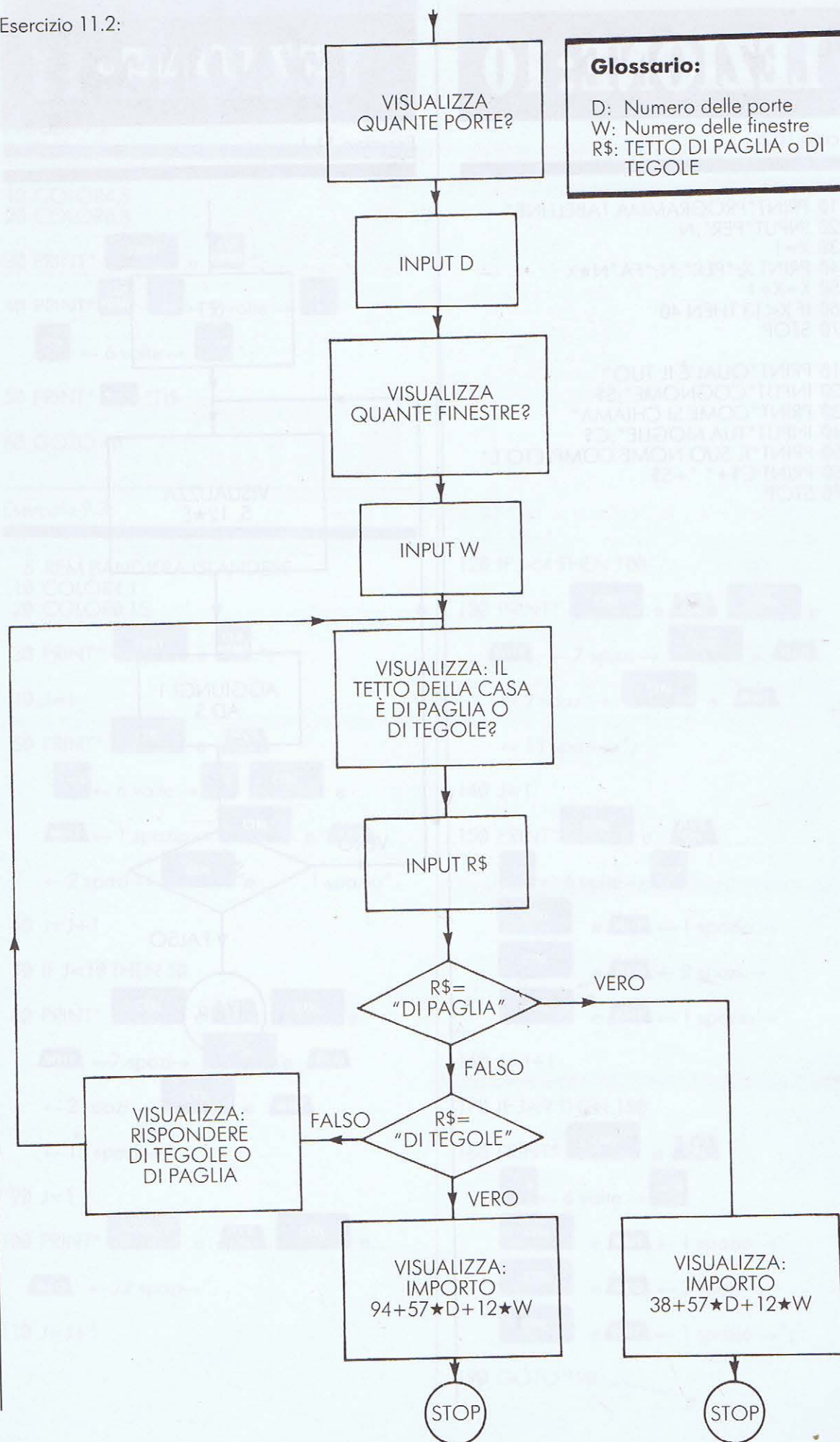
Esercizio 10.2:

a)
```
10 PRINT"PROGRAMMA TABELLINE"
20 INPUT"PER";N
30 X=1
40 PRINT X;"PER";N;"FA"N★X
50 X=X+1
60 IF X<13 THEN 40
70 STOP
```

b)
```
10 PRINT"QUAL'È IL TUO"
20 INPUT"COGNOME";S$
30 PRINT"COME SI CHIAMA"
40 INPUT"TUA MOGLIE";C$
50 PRINT"IL SUO NOME COMPLETO È"
60 PRINT C$+" "+S$
70 STOP
```

LEZIONE:11

Esercizio 11.1:

Esercizio 11.2:

```
 10 REM TASSE SULLA CASA
 20 PRINT"PROGRAMMA DI
     CLASSIFICAZIONE"
 30 INPUT"QUANTE PORTE";D
 40 INPUT"QUANTE FINESTRE";W
 50 PRINT"IL TETTO DELLA CASA E"
 60 PRINT"DI PAGLIA O"
 70 INPUT"DI TEGOLE";R$
 80 IF R$="DI PAGLIA" THEN 140
 90 IF R$="DI TEGOLE" THEN 160
100 PRINT"SI PREGA RISPONDERE"
110 PRINT"DI PAGLIA O"
120 PRINT"DI TEGOLE"
130 GOTO 50
140 PRINT"IMPORTO:";38+57★D+12★W
150 STOP
160 PRINT"IMPORTO:";94+57★D+12★W
170 STOP
```

Le risposte corrette ai tre problemi campione sono:

a) 95   b) 155   c) 364

LEZIONE: 12

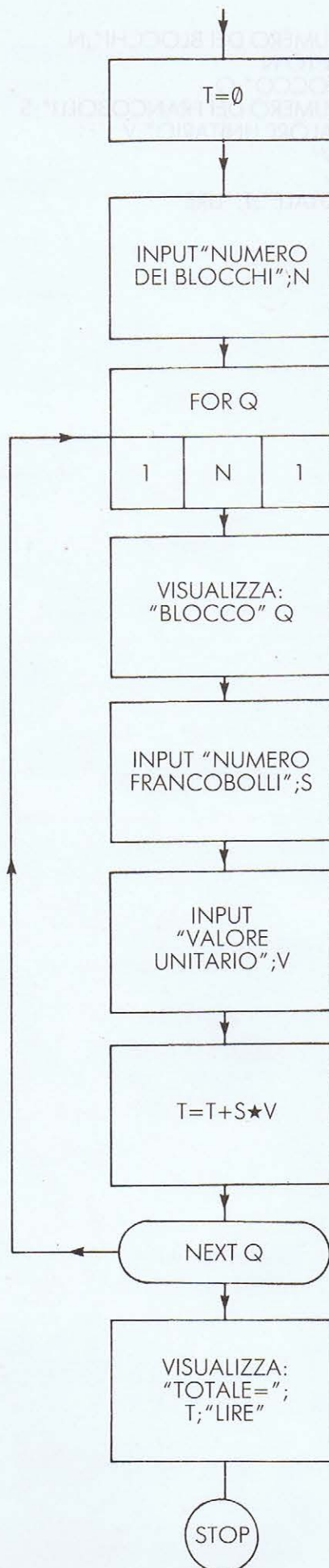
Esercizio 12.1:

```
10 RS=0
20 INPUT"NUMERO INNINGS";J
30 FOR Q=1 TO J
40 INPUT"PUNTEGGIO";S
50 RS=RS+S
60 NEXT Q
70 PRINT"MEDIA=";RS/J
80 STOP
```

entrambi gli ordini sono corretti

Esercizio 12.2:

Glossario:

N: Numero dei blocchi
S: Numero dei francobolli in un blocco
V: Valore di ogni francobollo in un blocco
T: Variabile di accumulo del totale
Q: Contatore dei blocchi

```
10 T=Ø
20 INPUT"NUMERO DEI BLOCCHI";N
30 FOR Q=1 TO N
40 PRINT"BLOCCO";Q
50 INPUT"NUMERO DEI FRANCOBOLLI";S
60 INPUT"VALORE UNITARIO";V
70 T=T+S★V
80 NEXT Q
90 PRINT"TOTALE";T;"LIRE"
100 STOP
```


LEZIONE:14

Esercizio 14.1:

Input: "SALDO PRECEDENTE"; B

VISUALIZZA: BATTERE ESTREMI DEGLI ASSEGNI USARE Ø PER CONCLUDERE

Input: "IMPORTO";V

V=Ø

Vero

Falso

PONI B=B–V

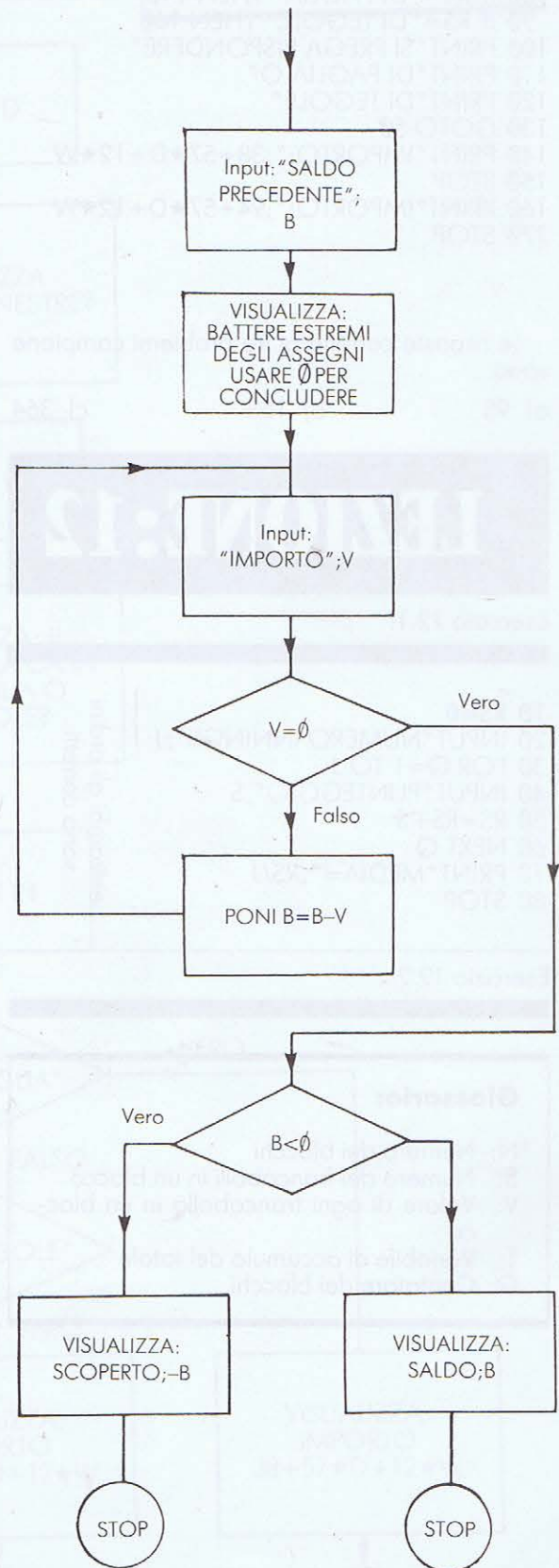
B<Ø

Vero

VISUALIZZA: SCOPERTO;–B

VISUALIZZA: SALDO;B

STOP

STOP

Glossario:

B: Saldo attuale
V: Ogni nuova transazione

```
 10 REM PROGRAMMA BANCARIO
 20 INPUT"SALDO PRECEDENTE";B
 30 PRINT"BATTERE ESTREMI DEGLI"
 40 PRINT"ASSEGNI.USARE 0
    PER CONCLUDERE"
 50 INPUT"IMPORTO";V
 60 IF V=0THEN 90
 70 B=B-V
 80 GOTO 50
 90 IF B<0THEN 120
100 PRINT"SALDO LIT.";B
110 STOP
120 PRINT"SCOPERTO"
130 PRINT"LIT.";-B
140 STOP
```

Esercizio 14.2:

(b)

VISUALIZZA INTESTAZIONE DESCRITTIVA

Input"PREZZO SUCCESSIVO" ;P

Stop

Vero

P=∅

Falso

P>= 12

Falso

T=P+⅓P

Vero

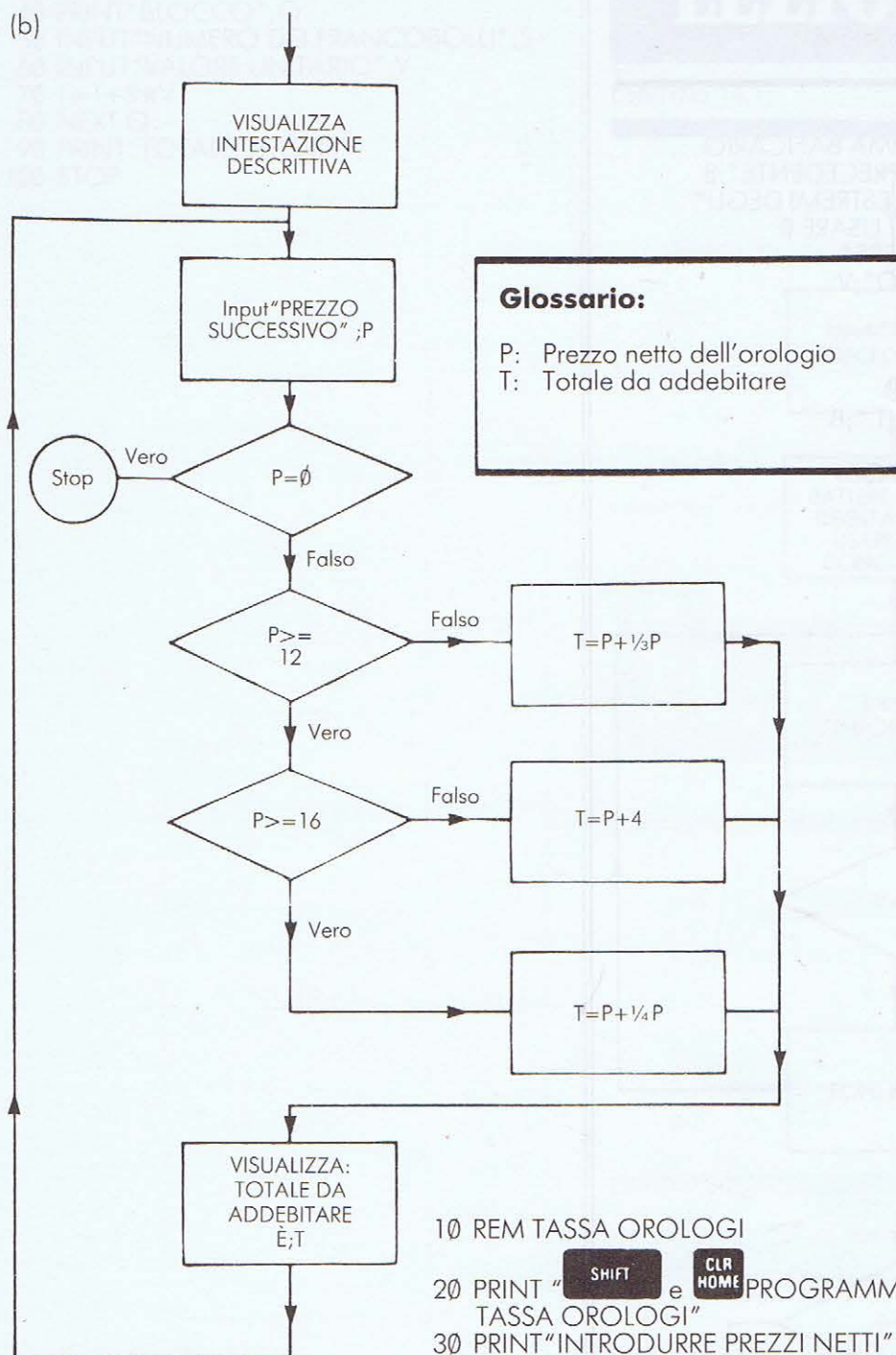
P>=16

Falso

T=P+4

Vero

T=P+¼P

VISUALIZZA: TOTALE DA ADDEBITARE È;T

Glossario:

P: Prezzo netto dell'orologio
T: Totale da addebitare


```
10 REM TASSA OROLOGI
20 PRINT " SHIFT e CLR HOME PROGRAMMA
   TASSA OROLOGI"
30 PRINT"INTRODURRE PREZZI NETTI"
40 PRINT"USARE ∅ PER CONCLUDERE"
50 INPUT"PREZZO SUCCESSIVO";P
60 IF P=∅ THEN 18∅
70 IF P>=12 THEN 1∅∅
80 T=P+(1/3)★P
90 GOTO 14∅
100 IF P>=16 THEN 13∅
110 T=P+4
120 GOTO 14∅
130 T=P+(1/4)★P
140 PRINT"IL TOTALE DA ADDEBITARE"
150 PRINT"È";T
160 PRINT
170 GOTO 5∅
180 STOP
```

Esercizio 14.2:

(c)

VISUALIZZA INTESTAZIONE DESCRITTIVA

Input L

L=Ø

Vero

VISUALIZZA: "INTRODURRE ALMENO UN NUMERO DIVERSO DA ZERO"

Falso

Input X

X=Ø

Vero

Falso

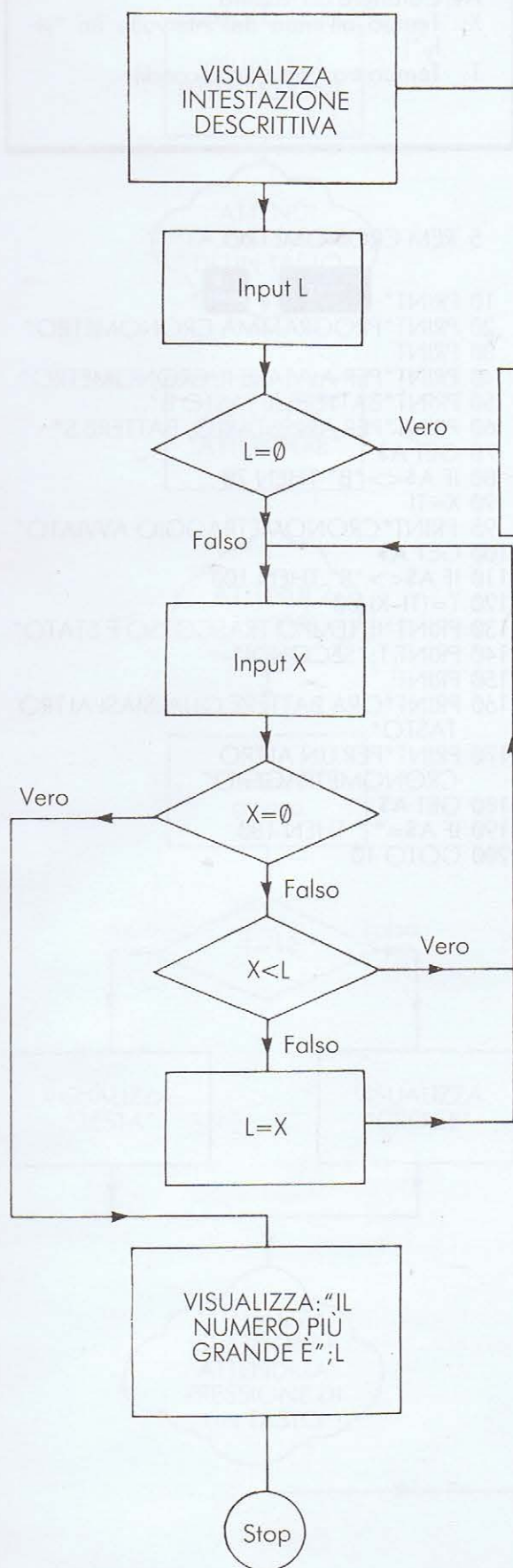
X<L

Vero

Falso

L=X

VISUALIZZA: "IL NUMERO PIÙ GRANDE È";L

Stop

Glossario:

L: Numero più grande incontrato fino ad ora
X: Prossimo numero da introdurre

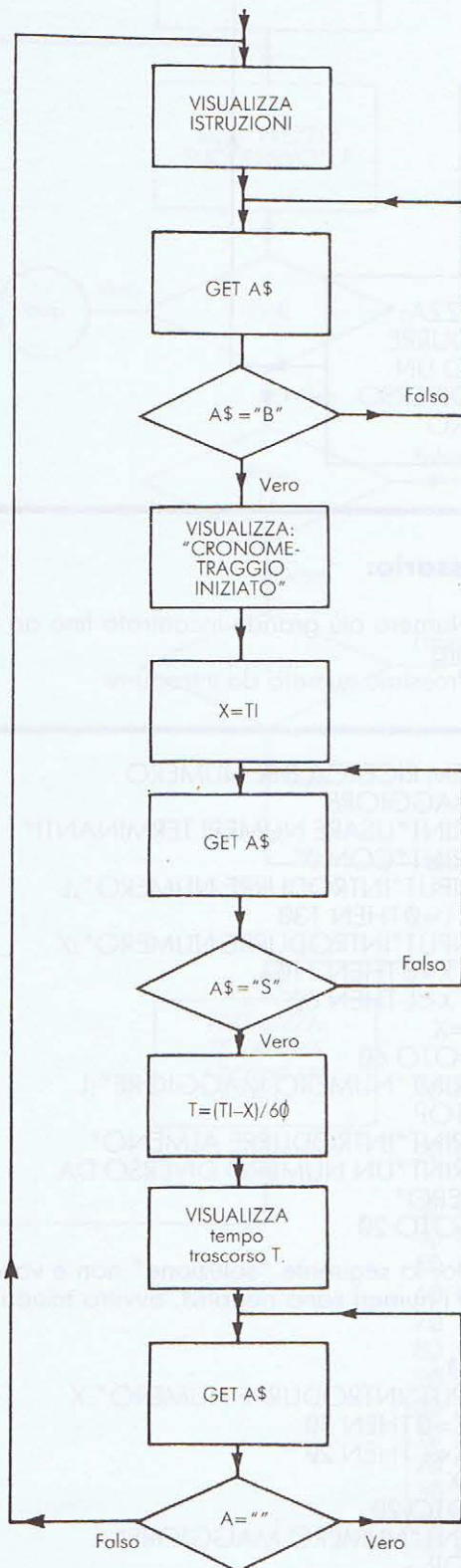
```
 1Ø REM RICERCA DEL NUMERO
      MAGGIORE
 2Ø PRINT"USARE NUMERI TERMINANTI"
 3Ø PRINT"CON Ø"
 4Ø INPUT"INTRODURRE NUMERO";L
 5Ø IF L=Ø THEN 13Ø
 6Ø INPUT"INTRODURRE NUMERO";X
 7Ø IF X=Ø THEN 11Ø
 8Ø IF X<L THEN 6Ø
 9Ø L=X
1ØØ GOTO 6Ø
11Ø PRINT"NUMERO MAGGIORE";L
12Ø STOP
13Ø PRINT"INTRODURRE ALMENO"
14Ø PRINT"UN NUMERO DIVERSO DA
      ZERO"
15Ø GOTO 2Ø
```

(Nota: la seguente "soluzione" non è valida se tutti i numeri sono negativi, ovvero minori di zero)

```
1Ø L=Ø
2Ø INPUT"INTRODURRE NUMERO";X
3Ø IF X=Ø THEN 7Ø
4Ø IF X<L THEN 2Ø
5Ø L=X
6Ø GOTO 2Ø
7Ø PRINT"NUMERO MAGGIORE";L
8Ø STOP
```

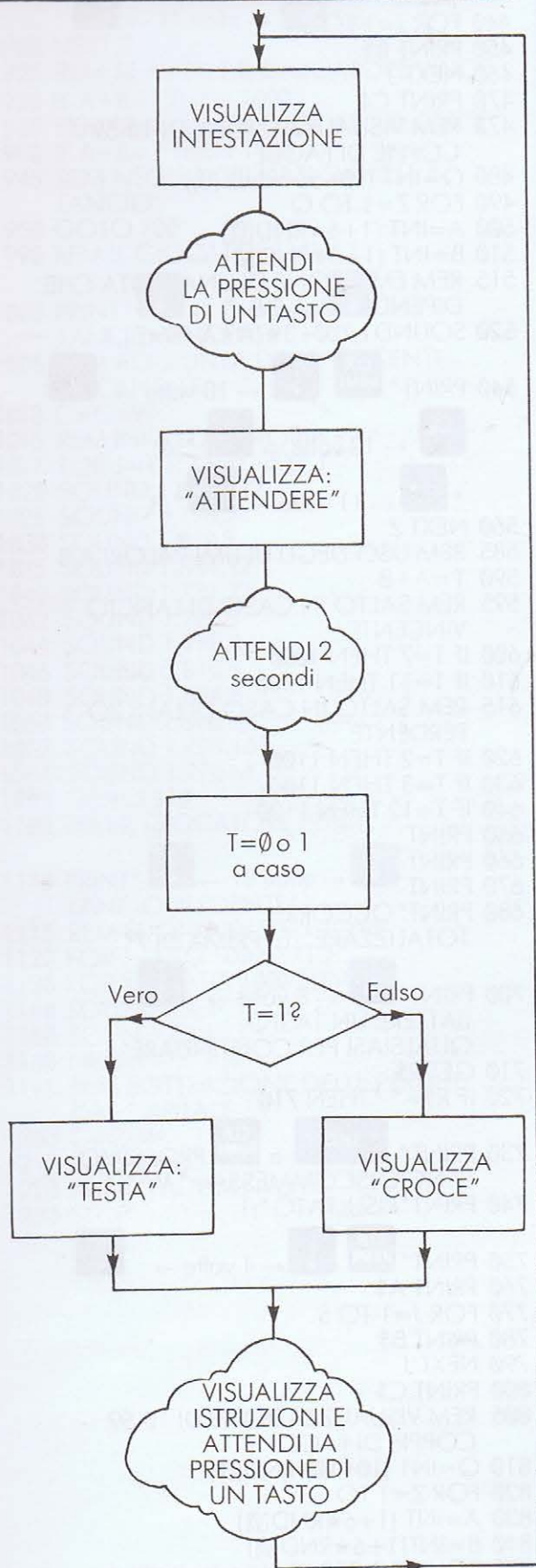
Scoprire il motivo per cui questo programma non funziona nel caso indicato.

LEZIONE:15

Esercizio 15.2 (1):

Glossario:

A$: Carattere da tastiera
X: Tempo all'inizio dell'intervallo (in "jiffy")
T: Tempo trascorso (in secondi)

```
5 REM CRONOMETRO
10 PRINT" SHIFT e CLR HOME "
20 PRINT"PROGRAMMA CRONOMETRO"
30 PRINT
40 PRINT"PER AVVIARE IL CRONOMETRO"
50 PRINT"BATTERE IL TASTO B"
60 PRINT"PER ARRESTARLO, BATTERE S"
70 GET A$
80 IF A$<>"B" THEN 70
90 X=TI
95 PRINT"CRONOMETRAGGIO AVVIATO"
100 GET A$
110 IF A$<>"S" THEN 100
120 T=(TI–X)/60
130 PRINT"IL TEMPO TRASCORSO È STATO"
140 PRINT T;"SECONDI"
150 PRINT
160 PRINT"ORA BATTERE QUALSIASI ALTRO
    TASTO"
170 PRINT"PER UN ALTRO
    CRONOMETRAGGIO"
180 GET A$
190 IF A$=" " THEN 180
200 GOTO 10
```

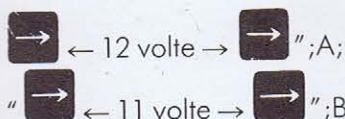
Esercizio 15.2 (2):

Glossario:

S$: Carattere da tastiera
M: Utilizzato nel "loop" di attesa (2 secondi)
T: Ø (per croce) o 1 (per testa)

```
10 REM LANCIO DELLA MONETA
20 PRINT" SHIFT e CLR HOME "
30 PRINT"BATTERE UN TASTO QUALSIASI"
40 PRINT"PER LANCIARE LA MONETA"
50 GET S$
60 IF S$=" "THEN 50
70 PRINT"ATTENDERE!"
80 PRINT
90 FOR M=1 TO 2000
100 NEXT M
110 T=INT (0+2★RND(0))
120 IF T=1 THEN 150
130 PRINT"CROCE"
140 GOTO 160
150 PRINT"TESTA"
160 PRINT
170 PRINT"BATTERE UN TASTO QUALSIASI
    PER"
180 PRINT"PROCEDERE"
190 GET S$
200 IF S$=" "THEN 190
210 GOTO 20
```

Esercizio 15.4:

GIOCO DEI DADI PLUS/4

```
  5 REM DADI
 1Ø REM 7
 3Ø PRINT"[SHIFT] e [CLR HOME] [CTRL]
    e [1 BLK] "
 4Ø PRINT"QUESTO GIOCO SI GIOCA
    CON"
 5Ø PRINT"DUE DADI. PRIMA
    SCOMMETTERE E QUINDI
 6Ø PRINT"LANCIARE. SE SI OTTIENE UN
    PUNTEGGIO DI 7 O 11,
 7Ø PRINT"SI VINCE. SE SI OTTIENE 2, 3 O
    12,
 8Ø PRINT"SI PERDE. SE SI OTTIENE
    QUALSIASI ALTRO NUMERO,
 9Ø PRINT"LA VITTORIA O LA PERDITA
    NON VENGONO DECISE SUBITO:
1ØØ PRINT"SI CONTINUA A LANCIARE
    FINCHÈ
11Ø PRINT"NON SI OTTIENE LO STESSO
    PUNTEGGIO REALIZZATO LA PRIMA
12Ø PRINT"VOLTA (E SI VINCE)
13Ø PRINT"O
14Ø PRINT"SI OTTIENE UN 7 (E SI PERDE).
15Ø PRINT
16Ø PRINT"BATTERE UN TASTO QUALSIASI
    PER CONTINUARE
24Ø GET A$
25Ø IF A$=" "THEN 24Ø
255 REM LE VARIABILI A$, B$, C$
    VENGONO IMPOSTATE PER
    DISEGNARE I DADI
26Ø A$="← 4 spazi →⌐‾‾‾¬
    ← 2 spazi →⌐‾‾‾¬"
27Ø B$="← 4 spazi →|← 3 spazi →
    |← 2 spazi →|← 3 spazi →|"
28Ø C$="← 4 spazi → ⌐‾‾‾¬
    ← 2 spazi →⌐‾‾‾¬
285 REM ACQUISIZIONE DEL CAPITALE
    INIZIALE
29Ø PRINT"[SHIFT] e [CLR HOME] "
3ØØ INPUT"CAPITALE INIZIALE";C
3Ø5 REM INIZIO SCOMMESSA
31Ø PRINT"BATTERE UN TASTO QUALSIASI
    PER INIZIARE LA SCOMMESSA"
33Ø GET R$
34Ø IF R$=" "THEN 33Ø
35Ø PRINT"IL CAPITALE ORA È";C
37Ø INPUT"QUANTO SCOMMETTI";W
38Ø IF W>ØTHEN 39Ø
385 PRINT"VALORE NON SIGNIFICATIVO"
387 GOTO 31Ø
39Ø IF W<=C THEN 42Ø
4ØØ PRINT"PUNTATA ECCESSIVA"
41Ø GOTO 31Ø
415 REM PREPARAZIONE PRIMO LANCIO
42Ø PRINT"[SHIFT] e [CLR HOME] [↓] [↓] [↓]
    PRIMO LANCIO
    (SCOMMESSA=";W;")"
43Ø PRINT"[CLR HOME] [↓] ← 7 volte → [↓] "
    ;A$
44Ø FOR J=1 TO 5
45Ø PRINT B$
46Ø NEXT J
47Ø PRINT C$
475 REM VISUALIZZAZIONE DI 1Ø-59
    COPPIE DI FACCE
48Ø Q=INT (1Ø+5Ø★RND (0))
49Ø FOR Z=1 TO Q
5ØØ A=INT (1+6★RND(Ø))
51Ø B=INT (1+6★RND(Ø))
515 REM EMISSIONE DI UNA NOTA CHE
    DIPENDE DA A E B
52Ø SOUND1,7ØØ+3★(A★A+B★B),4
54Ø PRINT"[CLR HOME] [↓] ← 10 volte → [↓]
    [→] ← 12 volte → [→] ";A;
    " [→] ← 11 volte → [→] ";B
56Ø NEXT Z
585 REM USO DEGLI ULTIMI VALORI A,B
59Ø T=A+B
595 REM SALTO IN CASO DI LANCIO
    VINCENTE
6ØØ IF T=7 THEN 1ØØØ
61Ø IF T=11 THEN 1ØØØ
615 REM SALTO IN CASO DI LANCIO
    PERDENTE
62Ø IF T=2 THEN 11ØØ
63Ø IF T=3 THEN 11ØØ
64Ø IF T=12 THEN 11ØØ
65Ø PRINT
66Ø PRINT
67Ø PRINT
68Ø PRINT"OCCORRE
    TOTALIZZARE";T;"PRIMA DI 7"
7ØØ PRINT"[↓] ← 8 volte → [↓]
    BATTERE UN TASTO
    QUALSIASI PER CONTINUARE"
71Ø GET R$
72Ø IF R$=" "THEN 71Ø
73Ø PRINT"[SHIFT] e [CLR HOME] PROSSIMO
    LANCIO(SCOMMESSA=";W;")"
74Ø PRINT"RISULTATO";T
75Ø PRINT"[CLR HOME] [↓] ← 4 volte → [↓] "
76Ø PRINT A$
77Ø FOR J=1 TO 5
78Ø PRINT B$
79Ø NEXT J
8ØØ PRINT C$
8Ø5 REM VISUALIZZAZIONE DI 1Ø-59
    COPPIE DI FACCE
81Ø Q=INT (1Ø+1Ø★RND(Ø))
82Ø FOR Z=1 TO Q
83Ø A=INT (1+6★RND(Ø))
84Ø B=INT(1+6★RND(Ø))
85Ø SOUND 1,7ØØ+3★(A★A+B★B),4
87Ø PRINT"[CLR HOME] [↓] ← 8 volte → [↓]
```

```
     →  ← 12 volte →  → ";A;
    " →  ← 11 volte →  → ";B
900 NEXT Z
925 REM SE A+B=T IL GIOCATORE VINCE
930 IF A+B=T THEN 1000
935 REM SE A+B=7 IL GIOCATORE PERDE
940 IF A+B=7 THEN 1100
945 REM NEGLI ALTRI CASI SI RIPETE IL
    LANCIO
950 GOTO 700
990 REM IL GIOCATORE VINCE
1000 PRINT" ↓ ← 7 volte → ↓
     LANCIO VINCENTE"
1005 REM AGGIUNTA DELLE VINCENTI
     AL CAPITALE
1010 C=C+W
1015 REM INNO DI VITTORIA
1017 FOR J=1 TO 500:NEXT J
1020 SOUND 1,834,332
1025 SOUND 1,798,24
1030 SOUND 1,810,8
1035 SOUND 1,834,32
1040 SOUND 1,739,32
1042 SOUND 1,770,8
1044 SOUND 1,798,8
1046 SOUND 1,810,8
1048 SOUND 1,834,8
1050 SOUND 1,810,16
1052 SOUND 1,798,16
1054 SOUND 1,770,64
1095 GOTO 310
1100 REM IL GIOCATORE PERDE
1110 PRINT" ↓ ← 10 volte → ↓
     LANCIO PERDENTE"
1115 REM SUONO DI DERISIONE
1120 FOR J=1 TO 500:NEXT J
1130 FOR X=800 TO 1000 STEP 4
1140 SOUND 1,X,1
1150 SOUND 1,X+23,1
1160 NEXT X
1195 REM SOTTRAZIONE DELLE PERDITE
     DAL CAPITALE
1200 C=C-W
1210 IF C>0 THEN 310
1220 PRINT"FALLIMENTO"
1230 STOP
```

APPENDICE A
SOLUZIONE AI PROBLEMI

Esempio 1:

```
10 INPUT V
20 INPUT R
30 PRINT"A"=;V↑2/R
40 STOP
```

Esempio 2:

```
10 PRINT"X FORMULA"
20 FOR X=0 TO 2 STEP 0.2
30 PRINT X;1/(1+X↑2)
40 NEXT X
50 STOP
```

Esempio 3:

```
10 PRINT"INTRODURRE I TRE LATI"
20 INPUT"A";A
30 INPUT"B";B
40 INPUT"C";C
50 S=(A+B+C)/2
60 X=S★(S-A)★(S-B)★(S-C)
70 IF X<0 THEN 100
80 PRINT"L'AREA È";SQR(X)
90 STOP
100 PRINT"QUESTI NON SONO I"
110 PRINT"LATI DI UN TRIANGOLO"
120 STOP
```

Glossario:

A,B,C: Tre lati di un triangolo
S: Semi-perimetro
X: Quadrato dell'area (se presente)

Esempio 4:

```
10 REM VERSIONE LEGGERMENTE PIÙ
   VELOCE"
20 INPUT"VALORE MASSIMO";H
30 FOR N=3 TO H STEP 2
40 Q=SQR(N)
50 FOR J=2 TO Q
60 IF N/J=INT (N/J) THEN 90
70 NEXT J
80 PRINT N;
90 NEXT N
100 STOP
```

APPENDICE C

Messaggi di errore

Questa lista comprende gli errori che si possono verificare usando le funzioni BASIC descritte in questo corso. Altri errori possono insorgere se si eseguono programmi più avanzati.

Division by Zero (divisione per zero)

La divisione per zero non è ammessa. L'errore può insorgere in comandi come

10 A=5/0

o

20 B=Q/(J–J)

Extra ignored (elementi ignorati)

Se si battono troppi elementi di informazione (numeri o stringhe) in risposta ad un comando INPUT, gli elementi in più verranno ignorati. Il programma non si arresta.

Illegal quantity (quantità illecita)

Un numero utilizzato in un comando è troppo grande (o troppo piccolo). Per esempio, qualsiasi numero che venga inserito in una locazione deve essere compreso fra 0 e 255.
Questo errore può insorgere in comandi come ad esempio:

SOUND10,5,37

o

COLOR300.2

Load error (errore di caricamento)

Il programma non viene caricato correttamente dal registratore a cassetta o dal floppy disk. Se si utilizza un registratore a cassetta, provare a pulire la testina. Oppure, può darsi che il programma non sia stato registrato correttamente all'origine o che il nastro sia stato danneggiato dalla vicinanza di una campo magnetico.

Next without for (Next senza For)

Il programma contiene una struttura FOR-NEXT errata.

Out of memory (memoria esaurita)

Il computer ha esaurito lo spazio disponibile in memoria. Ciò si verifica solo con programmi molto lunghi o che utilizzano grandi quantità di dati.

Redo from Start (ricominciare dall'inizio)

Se un comando INPUT richiede che venga battuto un numero in risposta ed invece si batte qualche cosa di diverso, il computer visualizza questo messaggio, e consente di effettuare un nuovo tentativo.

String too long (stringa troppo lunga)

Una stringa formata per concatenazione è più lunga di 255 byte.

Syntax error (errore di sintassi)

Un "comando" ha violato le regole della grammatica BASIC. Possibili cause sono parentesi non corrispondenti, parole chiave con ortografia errata o elementi di espressioni ordinati in maniera non corretta.

Type Mismatch (corrispondenza di tipo errata)

Ciò significa che un numero è stato usato al posto di una stringa o viceversa.

Verify Error (errore di verifica)

Il processo di verifica non è riuscito. Effettuare un nuovo tentativo di salvare il programma.

INDICE ALFABETICO

INTRODUZIONE AL BASIC PARTE 1

NASTRO 1

TESTCARD
HANGMAN
SPEEDTYPE
UNIT3QUIZ
UNIT4DRILL
UNIT5QUIZ

NASTRO 2

SENTENCES
UNIT7QUIZ
UNIT8PROG
UNIT9QUIZ
UNIT10QUIZ
UNIT11PROG
UNIT12QUIZ
TUNE
HEADS
REACTION
CRAPS

I programmi sono registrati su tutti i due lati dei nastri.